DOCUMENT RESUME

ED 111 337	IR 002 341
AUTHOR	Gunwaldsen, Roger L.
TITLE	CHARGE Image Generator: Theory of Operation and Author Language Support. Technical Report 75-3.
INSTITUTION	Human Resources Research Organization, Alexandria, Va.
SPONS AGENCY	Army Research Inst. for the Behavioral and Social Sciences, Arlington, Va.
REPORT NO	HumRRO-TR-75-3
PUB DATE	May 75
NOTE	137p.
EDRS PRICE	MF-$0.76 HC-$6.97 Plus Postage
DESCRIPTORS	*Computer Assisted Instruction; *Computer Graphics; Computer Oriented Programs; Computer Programs; Computers; *Display Systems; Educational Technology; Information Processing; Information Storage; Information Systems; Military Training; *On Line Systems; *Programing Languages; Teaching Machines; Three Dimensional Aids
IDENTIFIERS	*CHARGE; Color Halftone Area Graphics Environment

ABSTRACT

The image generator function and author language software support for the CHARGE (Color Halftone Area Graphics Environment) Interactive Graphics System are described. Designed initially for use in computer-assisted instruction (CAI) systems, the CHARGE Interactive Graphics System can provide graphic displays for various applications including equipment design, troubleshooting, and mathematics. It can achieve color, gray level, and three-dimensional (3-D) perspectives. This report describes the system's procedure for representing a complex 3-D world, forming a colored perspective of that world, and then encoding the perspective in a format for tranmission to and display on a CHARGE terminal. Examples of CHARGE inputs and the system's resulting graphic displays together with flowcharts of the process are provided. (Author/DGC)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

Technical
Report
75-3

HumRRO-TR-75-3

HumRRO

ED111337

CHARGE Image Generator: Theory of Operation and Author Language Support

Roger L. Gunwaldsen

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

HUMAN RESOURCES RESEARCH ORGANIZATION
300 North Washington Street • Alexandria, Virginia 22314

Approved for public release; distribution unlimited.

May 1975

IR 002 341

Prepared for

U.S. Army Research Institute for the
Behavioral and Social Sciences
1300 Wilson Boulevard
Arlington, Virginia 22209

2

The Human Resources Research Organization (HumRRO) is a nonprofit corporation established in 1969 to conduct research in the field of training and education. It is a continuation of The George Washington University Human Resources Research Office. HumRRO's general purpose is to improve human performance, particularly in organizational settings, through behavioral and social science research, development, and consultation. HumRRO's mission in work performed under Contract DAHC19-73-C-0004 with the Department of the Army is to conduct research in the fields of training, motivation, and leadership.

The contents of this paper are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

Published
May 1975
by
HUMAN RESOURCES RESEARCH ORGANIZATION
300 North Washington Street
Alexandria, Virginia 22314

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER
HumRRO-TR-75-3

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
CHARGE IMAGE GENERATOR: Theory of Operation and Author Language Support

5. TYPE OF REPORT & PERIOD COVERED
Technical Report

6. PERFORMING ORG. REPORT NUMBER
Technical Report 75-3

7. AUTHOR(S)
Roger L. Gunwaldsen

8. CONTRACT OR GRANT NUMBER(S)
DAHC19-73-C-0004

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Human Resources Research Organization (HumRRO)
300 North Washington Street
Alexandria, Virginia 22209

10. PROGRAM ELEMENT PROJECT TASK AREA & WORK UNIT NUMBERS
63101A; 2Q763731A734-00; 001

11. CONTROLLING OFFICE NAME AND ADDRESS
U.S. Army Research Institute for the Behavioral and Social Sciences
1300 Wilson Boulevard, Arlington, Virginia 22209

12. REPORT DATE
May 1975

13. NUMBER OF PAGES
132

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Research performed by HumRRO Eastern Division, Alexandria, Virginia, under Work Unit CATALIST.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

CHARGE Image-Generator
CHARGE Interactive Graphics System
CHARGE terminal
Colored image
CRT
Graphics terminal
Hardware
Image encoding
Instructional systems
Projection processor
Visible Surface processor

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report documents the image generator function and author language software support for the CHARGE Interactive Graphics System. Designed initially for use in instructional systems, the CHARGE Interactive Graphics System can provide considerable and versatile graphic displays for many applications (including equipment design, troubleshooting and mathematics). While being cost-competitive, the CHARGE system can achieve color, gray level,

(Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. *(Continued)*

three-dimensional (3-D) perspectives, and rapid updating. Perspective views can be simply, rapidly, and economically generated from a (3-D) xyz-coordinate, dynamically changing, "real world" description. This report describes from a functional viewpoint, the CHARGE system's procedure for representing a complex 3-dimensional world, forming a colored perspective view of this 3-dimensional world, and then encoding the perspective in a format for transmission to and display on a CHARGE terminal.

MILITARY PROBLEM

The combination of shrinking financial resources and the prospects of a smaller, all-volunteer Army will increase both the demands made on Army personnel and the importance of the individual soldier. There will be a greater need for more effective and efficient training, adequate to the task of providing an increasing number of complex skills to widely differing students, while using fewer skilled instructors.

The most promising approach available to meet these new training demands is computer-administered instruction (CAI), if it is developed as a comprehensive, total system.

The goal of Project IMPACT-CATALIST is to provide the Army with an effective, efficient, and economical CAI system in a total system framework. To be effective, the system should maximize the achievement of the students and the instructors to a greater extent than is possible in the traditional classroom; to be efficient, it should provide maximum productivity per unit time on the part of instructors, administrators, and students; and to be economical, the cost and resources must not exceed those of a comparable effective non-CAI instructional system.

DEVELOPMENT PROBLEM AND APPROACH

The objective of the CHARGE Interactive Graphics System effort was to expand the visual link between the CAI system and the student. It was believed that the overall effectiveness of the training process could be increased if the CAI System were provided with the ability to logically construct and present to the student graphic images of the most general nature.

Accordingly, at the outset the CHARGE development effort considered graphic display of dynamic 3-dimensional real and abstract worlds. A system able to handle such problems can easily handle subset problems such as display of alphanumeric text, line drawings, and 2-dimensional graphics.

RESULTS

A powerful interactive graphics system, called CHARGE, has been designed. The CHARGE system has three components:

(1) The host central computing system that manages the CAI system and maintains a symbolic representation of the 3-dimensional world under consideration.

(2) An image generator functional unit, time-shareable among the CAI users, that translates the 3-dimensional world description and student responses into an encoded, colored, pictorial image.

(3) A graphics terminal that buffers the encoded image at low cost and displays it to the student as a colored pictorial image.

A detailed description of the 3-dimensional world description and image generator function is provided in this report. A description of the CHARGE terminal is given in HumRRO Technical Report 74-26, *CHARGE Interactive Graphics System Terminal: Theory of Operation*, December 1974.

PREFACE

This report describes the results of the development by the Human Resources Research Organization of the CHARGE Interactive Graphics System image generator and author language support. The image generator is described from a theoretical and detailed functional processing point of view.

The work was begun at HumRRO Division No. 1 (Systems Operations), Alexandria, Virginia, under Project IMPACT in 1971 and has continued under Work Unit CATALIST. Dr. J. Daniel Lyons is Director of the Division (now the HumRRO Eastern Division), and Dr. Robert J. Seidel is Program Director, Instructional Technology Group.

Members of the CHARGE System R&D Team were Dr. Ronald J. Swallow, Principal Investigator, Mr. Roger L. Gunwaldsen, and Mr. William G. Underhill.

Detailed description of the CHARGE terminal is given in the report: *CHARGE Interactive Graphics System Terminal: Theory of Operation*, Technical Report 74-26, December 1974.

The work described in this report was conducted for the Department of the Army under Contract 19-73-C-0004. Computer-assisted instruction research is conducted under Army Project 2Q763731A734.

Meredith P. Crawford
President
Human Resources Research Organization

8

CONTENTS

Page

Summary 1

Preface 3

Introduction to the CHARGE System 11
- Background 11
- System Overview 11

Section

1 CHARGE Image Generator 19
- Overview 19
- World Description and View Parameters 20
  - Introduction 20
  - 3-Dimensional Space R^3 20
  - World Description: Observer 21
  - World Description: Light Sources 22
  - World Description: Object Definition Requirements 22
  - World Description: Objects, Atoms, and Transformations 25
  - World Description: Atom Data (Definition) 28
  - Final Atom Data 35

2 Projection Processor 36
- Introduction 36
  - Input Data 36
  - Observer 36
  - Light Sources 37
  - Atoms 39
  - Processing Steps 41
  - Initial Processing 42
  - Atom Projection Loop 45
  - Window Clipping: X and Y 58
  - Window Clipping: Z 67
  - Formulation of Atom Edge in R^2 72
  - Y-Ordered Output Edge Set 76

3 Visible Surface Processor 78
- Introduction 78
- Input Data 78
- Output Data 79
- Overview of Scanline Operation 80
- Visible Surface Algorithm 83
- Visible Edge Encoder 100
- Vertical Skip-Ahead 106

Appendices Page

A Example of a Non Planar Surface 117
B Logarithmic Brightness Parameters 130
C Errors in Visible Edge Position and Brightness at Terminal 131
D CHARGE Commands 133

LIST OF ILLUSTRATIONS

Figures

1 The CHARGE System 12
2 Object File Structure 13
3 CHARGE System Functional Flowchart 15
4 User In His 3-Dimensional World 17
5 Projected and Clipped Atoms 17
6 Encoded Edge Image 17
7 Image Generator Processing Steps 19
8 Standard Coordinate System 20
9 Observer Viewing Orientation in R^3 22
10 Pyramid of Vision 23
11 Standard Orientation of Atoms in "Table" 26
12 "Table" (Original Specification) 27
13 "Table" ("Tableleg" Modified) 28
14 Unit Volume Cube 30
15 Planar Surface S 31
16 Edge Between Planar Surfaces S_L and S_R 31
17 Smooth-Surface S_{sm} and Flat-Surface S_{fl} 32
18 Brightness Within Planar Surface Perimeter 34
19 Observer in R^3 38
20 Brightness at Point $\overline{X}$ Due to Point Source at Point $\overline{X}_P$ 39
21 Projection Processor Processing Steps 41
22 World After $[T'_w]$ Transformation 43
23 World After $[T_w]$ Transformation 43
24 Projection of Point $\overline{P}$ onto the Viewing Window 47
25 Projection of a Straight Line Edge 48
26 Projected Atom Planar Surface S 49
27 Projected Edge With P'_2 above P'_1 51
28 Projected Edge With P'_1 above P'_2 51
29 Projection of One Point on Edge and Arbitrary Point on Plane S 51
30 Depth at Arbitrary Point $\overline{P}'$ 54
31 Projection of an Edge With Two Brightness Normals 56
32 Translated Axes X and Y With Corresponding Viewing Window 59
33 Effect of $W_x/W_y \neq 4/3$ $(R_x/R_y \equiv W_x/W_y)$ 59

Figures *(Cont.)* Page

34 Unclipped Projected Edges 64
35 Edges Clipped in Y 64
36 Edges Unclipped in X 66
37 Edges Clipped in X 66
38 Intersection of Plane ABCD and Viewing Window Plane 68
39 Intersection of General Planar Surface With $Z=Z_w$ Plane 69
40 Common Point to Both S and $Z = Z_{vw}$ 70
41 Y Intercept Data Structure: $y_1>y_2>y_3>y_4$ 71
42 Projected Edge on Discrete Raster 73
43 Projected Edge Starting and Ending on Discrete Scanlines 74
44 Projected Edges Ending on Discrete Scanlines 75
45 Visible Surface Processor Functional Overview 81
46 Edges Defining Image 84
47 Edges in Block 1 84
48 Edges in Block 2 84
49 Visible Surface Algorithm 85
50 Beam Box Format 86
51 Edge E in i^{th} X Range 87
52 Change Table Entry 87
53 Delayed Removal of Surface S at X = i+1 on Scanline j 88
54 Scanline Partitioning 90
55 Video Partition Format 91
56 Intersection of Visible Surfaces 92
57 Pending Brightness Table Form 93
58 Projected Plane and Cylinder 94
59 Close-Up View of Plane and Cylinder 95
60 S_3:S_1 Ambiguous Scanline Partition 96
61 Multiple Edges Referencing S_1 on Left 96
62 Choice of Edges at X = i 97
63 Setting Output Mode 97
64 Three Step Skip-Ahead 99
65 Current Visible Edge Table Entry Format 100
66 Current Edge Block Table Entry Format 101
67 Visible Edge Encoder 103
68 Projected Edge; Edge at Visible Edge Encoder; Edge at CHARGE Terminal 105
69 Visible Edge Encoding on Scanline Y=j 106
70 Order of Edges for Block Depicting a Cube 107
71 Empirical Skip-Ahead Decision Rule 110
72 Probability Density Functions for Vertical Skip-Ahead 111
73 Vertical Scanline Processing 113

Figures *(Cont.)*

Page
A-1 Right Cylinder 117
A-2 Right Cylinder Represented by n=8 Planar Surfaces 118
A-3 $e(\phi) = e(\overline{x}) = \| \overline{x} - \hat{x} \|$ 119
A-4 Reduction of $e_{N_{max}}$ by Increasing the Number of Cylinder Facets 120
A-5 Typical Brightness Reflectivity Function 122
A-6 Smooth Shading Equivalent 122
A-7 Precise and Approximating Brightness Functions 123
A-8 Brightness Error $e_b(-\pi/n, n, \phi_1)$ 124
A-9 Brightness Function Derivatives 126
A-10 Variation of λ_{max} With (B_f/B_o) 128
A-11 Deviation Between $B(\lambda)$ and $\hat{B}(\lambda)$ 129

Tables

A-1 Brightness Approximation Maximum Errors 125

CHARGE Image Generator Theory of Operation and Author Language Support

INTRODUCTION TO THE CHARGE SYSTEM

BACKGROUND

The objective of Project IMPACT-CATALIST is to evolve a series of prototype systems of computer-administered instruction (CAI) in order to produce a total CAI system that is effective, efficient, and cost-effective for operational use in a training/instructional environment. The total prototype system includes four main components:

(1) Hardware—the computer, student stations, and related equipment.

(2) Software—the computer programing systems that control operation of the hardware.

(3) Courses of instruction—the actual content and logic of courses administered by the computer.

(4) Instructional Decision Models (IDMs)—the rules and strategies by which specific course content is provided to an individual student.

As part of the Work Units IMPACT and CATALIST, HumRRO conducted a system design study which led to the development of an interactive graphics system called CHARGE (Color Halftone Area Graphics Environment). Designed initially for use in instructional systems, the CHARGE system can provide considerable and versatile graphic displays for many applications (including equipment design, troubleshooting, etc.). While being cost-competitive, the CHARGE system can achieve color, gray level, three-dimensional perspectives, and rapid updating. Perspective views can be simply, rapidly, and economically generated from a (3-dimensional) xyz-coordinate, dynamically changing, "real world" description.

SYSTEM OVERVIEW

The CHARGE system has three major components: a host central computing system, an image generator, and a set of user display terminals. A detailed description of the user display terminals is given in the IMPACT-CATALIST report: *CHARGE Interactive Graphics System Terminal: Theory of Operation.*[1] These three components are configured as in Figure 1.

---

[1] Ronald J. Swallow, *CHARGE Interactive Graphics System Terminal: Theory of Operation*, HumRRO Technical Report 74-26, December 1974.

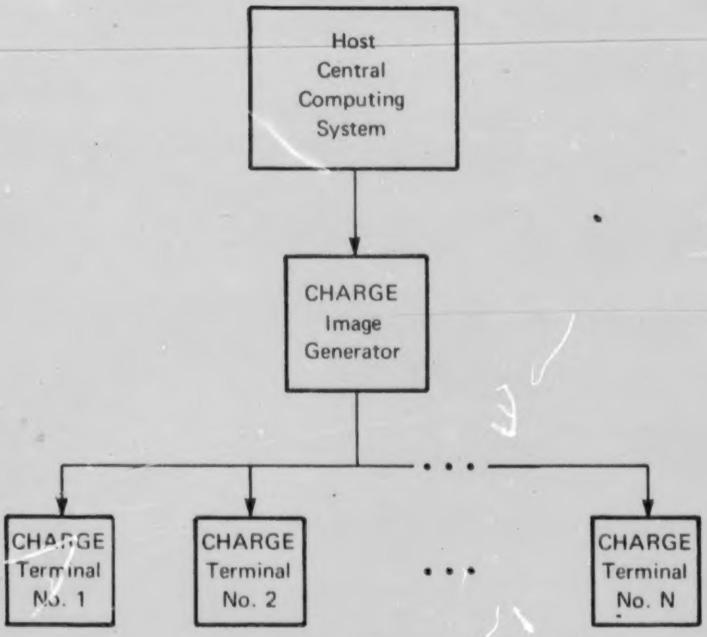


Figure 1 – **The CHARGE System**

The host central computing system interacts with each user as he creates his 3-dimensional "world," a description of a 3-dimensional region of space. The host does this by maintaining and manipulating two separate files that serve as the data base for the CHARGE system:

- The first file is the "atom" file, a low-level data base that defines a library of standard 3-dimensional shapes or building blocks such as a unit cube, unit sphere, and so forth. These atoms are generally not modified by the user; in fact, the user does not even have to know what data are used to represent an atom, he merely has to know each atom's standard orientation in space.
- The second file is the "object" file, a tree-structured file by which the user can create and name complex objects by defining them in terms of previously defined objects and atoms. The user does this by use of symbolic commands such as: "Create object A by adding objects B and C. Then create object D by adding objects A, B, and A, this time moved up 3 meters in y in the xyz space." The linkages between levels of object definition are "transformations" (such as, "move 3 meters in y"). The lowest level definition in any object is always a transformed atom. The object file structure for object D in the above example is given in Figure 2.

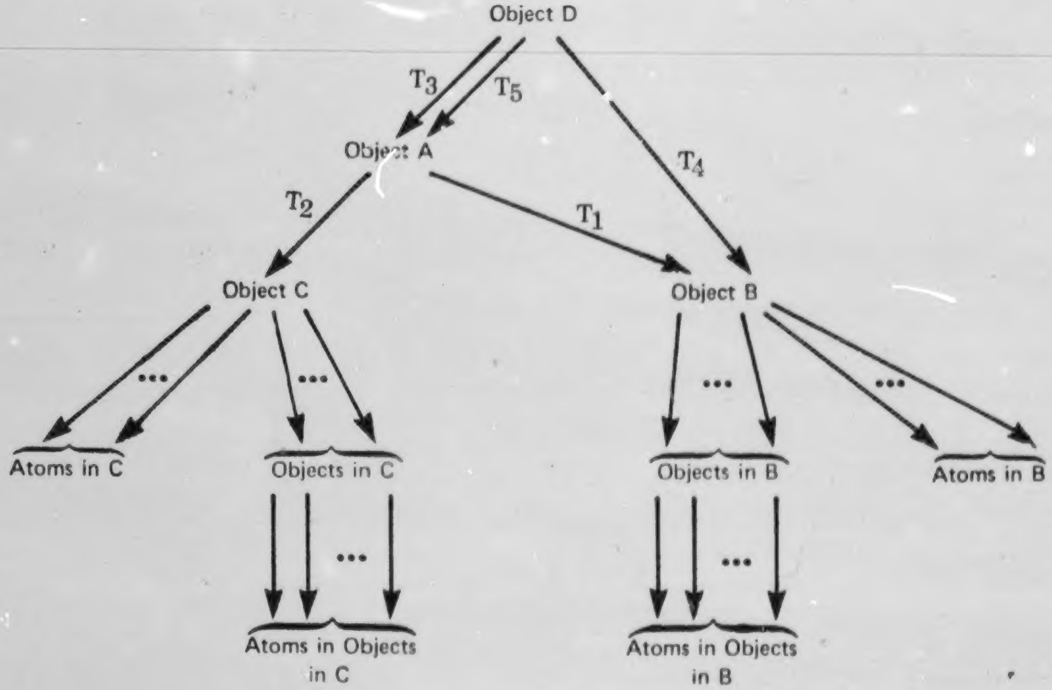


Figure 2 – **Object File Structure**

It should be noted that the data structures in the host central computing system do not represent the graphic view of the world; rather, they represent a data base from which an infinite number of graphical perspectives can be made merely by the command "Put me at (x,y,z) in this world and show me what I see." In fact, the host can put several users into the same world and generate views for each using the same data base. In this aspect, the host system is far more efficient in its file storage than it would be if it had to store "canned" (i.e., pre-computed) perspectives. Furthermore, the CHARGE system has the capability of generating a dynamical representation of this 3-dimensional world by giving the transformations and/or observer time varying parameters. The cost and performance of the host is shared by the N users.

The process of generating the perspective begins at the time the user gives the "show" command to the host. The host then initiates this process by first "compiling" the user's world. The compilation process accomplishes two tasks:

(1) Each atom that is symbolically referred to in the object file has its atom data retrieved from the atom file.

(2) Each transformation string linking an atom to the world is replaced with an equivalent composite transformation.

These two sets of data now define the user's symbolic world. There is one composite transformation for each occurrence of an atom in the user's symbolic description. The amount of actual atom data is related to the number of different atoms symbolically referred to by the user. For example, if the user's symbolic object file had 50 atoms, each one a Cube, the compiler would produce 50 composite transformations but the only atom data that would be retrieved is that of the atom Cube. These data along with the user's viewing orientation in his 3-dimensional world are then sent to the CHARGE image generator for perspective computation, thereby freeing the host for another user.

The top half of Figure 3 depicts each user's flow through the host computing system as he creates and views a 3-dimensional world. A typical user will spend most of his time in a "think" mode causing the host to loop through all of the "No" branches each time the user is polled. The top three boxes are entered infrequently compared to think time. When they are entered, they are typical interactive user requests requiring very small amounts of the host's resources.

The greatest load to the host computer is that of compiling a user's world, but this too is a low-frequency event. Furthermore, the demands on the host are still quite small. A world with 100 atoms, each with an average linkage depth of 4 levels would require approximately $100 \times 4 \times 27 \doteq 1.1 \times 10^4$ multiplications to produce the 100 composite transformations. If the host has a multiplication time of 3μsec/multiplication, this would be 33 msec. of computation, a fraction of the real time required to retrieve the user's atom and object files from disk storage. The computational load of computing the user's image is off-loaded onto the CHARGE image generator.

The CHARGE image generator is a processor located between the host central computing system, and the N user terminals. Its function is to translate each user's 3-dimensional world/viewer data into a high resolution encoded image that can be buffered locally at low cost at the user's terminal. Like the host, the image generator's cost and resources are shared by the system's N users. At the present time, the image generator function is being emulated in software, thereby allowing for experimentation in new algorithms before a hardware design is finalized.

The CHARGE image generator is really a two-stage processor. The compiled world data from the host enters the projection processor which projects each atom into a "perspective domain" and limits the resulting perspective domain data to those atom surfaces that would be visible to the user on his monitor if no other atom surfaces were present. The exact form of this perspective domain atom data is chosen to maximize the

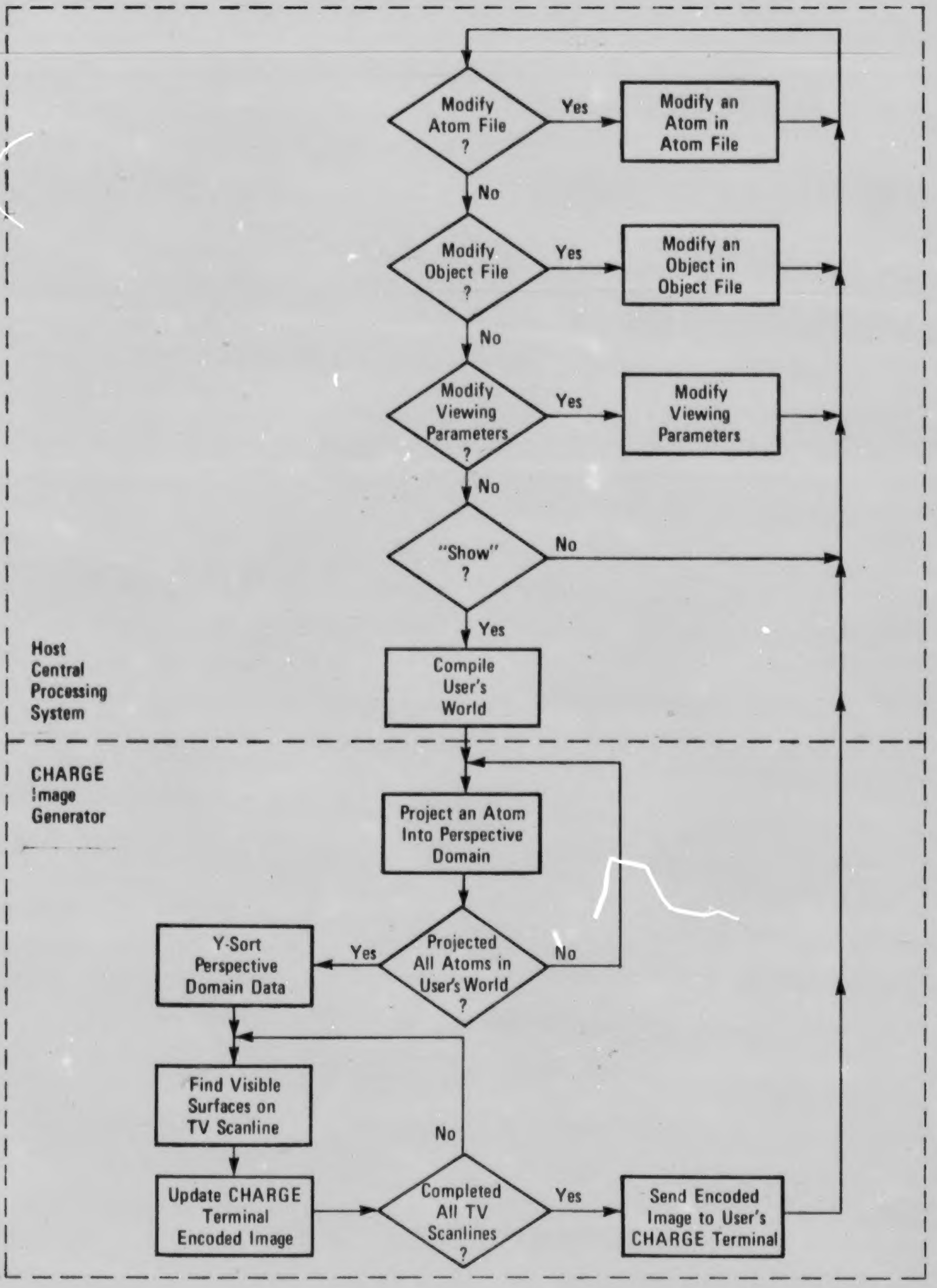

Modify Atom File ?
Yes
Modify an Atom in Atom File
No
Modify Object File ?
Yes
Modify an Object in Object File
No
Modify Viewing Parameters ?
Yes
Modify Viewing Parameters
No
"Show" ?
No
Yes
Compile User's World
Host Central Processing System
CHARGE Image Generator
Project an Atom Into Perspective Domain
Y-Sort Perspective Domain Data
Yes
Projected All Atoms in User's World ?
No
Find Visible Surfaces on TV Scanline
No
Update CHARGE Terminal Encoded Image
Completed All TV Scanlines ?
Yes
Send Encoded Image to User's CHARGE Terminal

Figure 3 – **CHARGE System Functional Flowchart**

rate at which the second stage processor, the visible surface processor, is able to determine which surface is actually visible at each (x,y) coordinate on the user's monitor. The visible surface processor also encodes the resultant image into a highly data-compressed format that can be efficiently transmitted to and economically buffered at the user's CHARGE terminal.

Once the encoded image has been transmitted to the user, the image generator is free to service another user. The two stages of the image generator are independent and if a second buffer for y-sorted perspective domain atom data is added, the projection processor can fill one y-sorted buffer with one user's data while the visible surface processor empties the other y-sorted buffer containing a second user's data.

Figure 4 demonstrates the operations of the image generator on an extremely simple case. The user has symbolically created a world consisting of just two atoms: a cube and a triangular plane. The host sends to the image generator the following data:

(1) The user's orientation
(2) Atom data for the standard cube
(3) A transformation orienting the standard cube in the 3-dimensional world
(4) Atom data for the standard triangular plane
(5) A transformation orienting the standard triangular plane in the 3-dimensional world.

The projection processor first projects the two atoms onto the user's "viewing window," the rectangular area corresponding to the user's monitor. Back surfaces are "clipped away." If any atom surfaces had projected outside of the viewing window rectangle, they too would have been clipped away. The resulting perspective domain data consisting of surfaces and edges is shown in Figure 5. Note that no "front surface" decisions have yet been made between the surface of the cube and the surface of the triangular plane.

The visible surface processor resolves conflicts between these overlapping surfaces and encodes the image into the edge form of Figure 6. Note that as conflicts between overlapping surfaces are resolved, intersection edges are automatically created.

Figure 6 also demonstrates the efficient data format at the CHARGE terminal. Each "visible edge" in Figure 6 defines a color and a brightness to its right side; the edges themselves are ordered in x such that for any scanline (y position) all of the edges to the right of Edge i have edge labels greater than i. Edge 1 is a special "marker" edge that flags to the CHARGE terminal decoder unit that there are data present. The image of Figure 6 contains 20 edges requiring that $20 \cdot 76 = 1520$ bits be buffered at the CHARGE

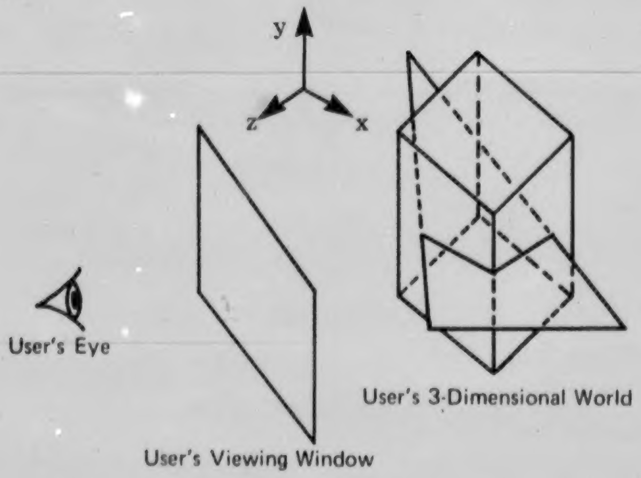


Figure 4 – **User in His 3-Dimensional World**

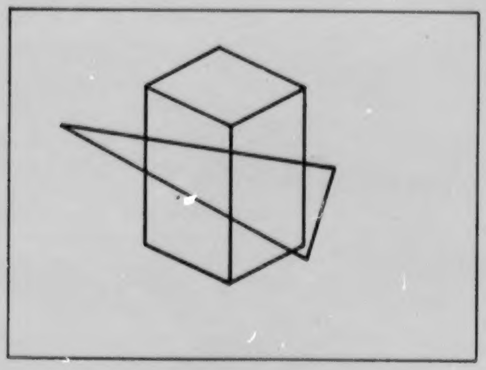


Figure 5 – **Projected and Clipped Atoms**

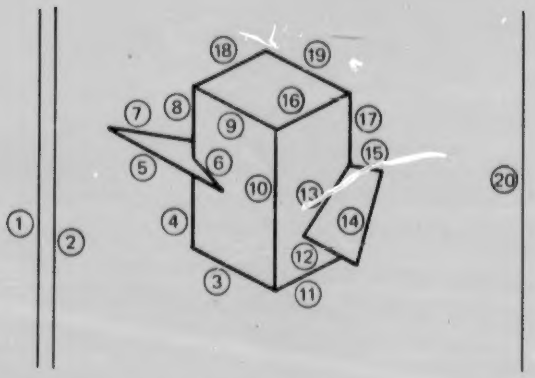


Figure 6 – **Encoded Edge Image**

terminal to produce a sharp, colored video image on a monitor with resolutions 1600 units horizontally by 1200 scanlines vertically (4:3 aspect ratio). If the same image were to be buffered as a video image with 9 bits of brightness and 12 bits of color information for each point, 20.7×10^7 bits would need to be buffered.

The CHARGE image generator and terminal also have the ability to "smooth-shade" curved surfaces; that is, curved surfaces are approximated by planar facets and the resulting edges have bits set instructing the CHARGE terminal decoder unit to linearly interpolate the brightness between successive edges, thereby giving them the appearance of a continuous, smooth, "round" surface with no edges within it. Thus, round surfaces such as spheres and cylinders appear round on the terminal.

Section 1

CHARGE IMAGE GENERATOR

OVERVIEW

The image generator may be thought of as a very fast, highly intelligent data translator located in the output data path from the central computing system to the display terminals. Input data from the central computing system are:

(1) World description: A complex, symbolic description of a region of finite 3-dimensional space.

(2) Viewing parameters: Definition of an observer with respect to this world.

Output data to the display terminal comprise a highly data-compressed representation of this world as seen by an observer with the specified viewing parameters. If the world description and/or the viewing parameters contain time-varying elements, the output becomes a real time sequence of these encoded images—that is, a movie of a changing world as seen by a moving observer.

The image generator is a two-step processor and operates as in Figure 7 on a symbolic world-observer pair at time instant t_i. The projection processor uses the observer's viewing parameters to map the symbolic 3-dimensional world description onto the observer's 2-dimensional display screen. Since projection maps each item in the world independently of the others, the projected world data must be further processed by the visible surface processor to determine which of the projected items are visible and to encode the resulting video image into a data-compressed form to be sent to the display terminal.

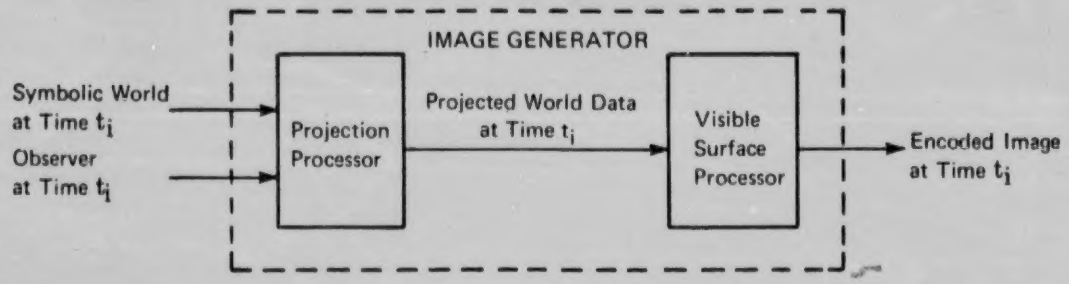


IMAGE GENERATOR
Symbolic World at Time t_i
Observer at Time t_i
Projection Processor
Projected World Data at Time t_i
Visible Surface Processor
Encoded Image at Time t_i

Figure 7 — Image Generator Processing Steps

This report on the image generator will contain three parts:

(1) Specification of a world description and viewing parameters
(2) Projection processor
(3) Visible surface processor

WORLD DESCRIPTION AND VIEWING PARAMETERS

Introduction

This section provides (a) a description of the 3-dimensional space, (b) the method of representing an observer within this space, (c) the method of representing light sources within this space, and (d) the method of representing 3-dimensional, automatically shaded objects within this space.

3-Dimensional Space R^3

All positions $\bar{x} = (x,y,x)^T = x\bar{i} + y\bar{j} + Z\bar{k}$ in the world are defined with respect to the righthanded rectangular coordinate system of Figure 8 where unit directional vectors $\bar{i} = (1,0,0)^T$, $\bar{j} = (0,1,0)^T$, and $\bar{k} = (0,0,1)^T$ satisfy the vector products:

$$\begin{array}{lll} \bar{i} \times \bar{i} = \bar{0} & \bar{j} \times \bar{i} = -\bar{k} & \bar{k} \times \bar{i} = \bar{j} \\ \bar{i} \times \bar{j} = \bar{k} & \bar{j} \times \bar{j} = \bar{0} & \bar{k} \times \bar{j} = -\bar{i} \\ \bar{i} \times \bar{k} = -\bar{j} & \bar{j} \times \bar{k} = \bar{i} & \bar{k} \times \bar{k} = \bar{0} \end{array}$$

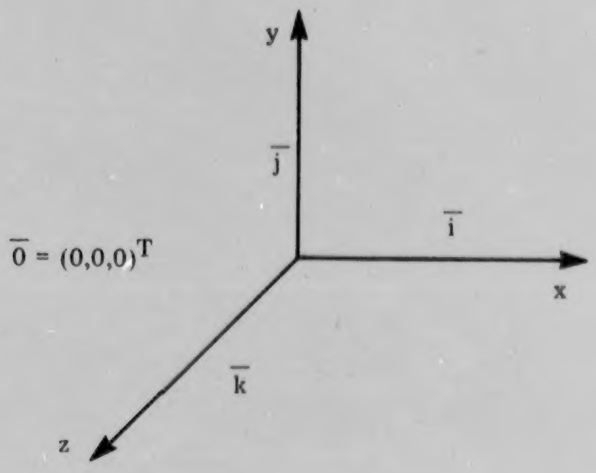


Figure 8 – **Standard Coordinate System**

The point $\bar{0} = (0,0,0)^T = 0\bar{i} + 0\bar{j} + 0\bar{k}$ is the origin of the world. The 3-dimensional space is defined to be finite and will be denoted by R^3.

World Description: Observer

The observer is completely specified by the set of viewing parameters $(X_W,Y_W,Z_W,X'_E,Y'_E,Z'_E,X_V,Y_V,Z_V,W'_X,W'_Y,MAG,B_o)$ which define an eye, a viewing window, and a point of observation. These parameters are pictorially represented in Figure 9. $\bar{V} = (X_V,Y_V,Z_V)^T$ defines the point of observation with respect to $\bar{0} = (0,0,0)^T$, the origin of R^3. This is the point toward which the observer directs his vision. The observer sees into R^3 only through a viewing window. This viewing window has a center located at $\bar{W} = (X_W,Y_W,Z_W)^T$ with respect to the origin of R^3, and lies in a plane orthogonal to the vector $\bar{V} - \bar{W}$.

$\bar{W}$ is the origin of a second rectangular axis coordinate system with axes x′,y′, and z′ and unit directional vectors $\bar{i}',\bar{j}'$, and $\bar{k}'$ defined so that:

$$\begin{array}{lll}
\bar{i}' \times \bar{i}' = \bar{0} & \bar{j}' \times \bar{i}' = -\bar{k}' & \bar{k}' \times \bar{i}' = \bar{j}' \\
\bar{i}' \times \bar{j}' = \bar{k}' & \bar{j}' \times \bar{j}' = \bar{0} & \bar{k}' \times \bar{j}' = -\bar{i}' \\
\bar{i}' \times \bar{k}' = -\bar{j}' & \bar{j}' \times \bar{k}' = \bar{i}' & \bar{k}' \times \bar{k}' = \bar{0}
\end{array}$$

$$\bar{k}' \cdot \frac{(\bar{V}' - \bar{W})}{\|\bar{V}' - \bar{W}\|} = -1$$

$$\bar{i}' . \bar{j} = \bar{0}$$

With respect to this primed coordinate system, the window size is W'_X horizontally by W'_Y vertically, and the y and y′ axes lie in a plane. The observer's eye is located at $\bar{E}' = (X'_E,Y'_E,Z'_E)^T$ with respect to the primed axes. If $X'_E = Y'_E = 0$, the eye looks at $\bar{V} = (X_V,Y_V,Z_V)^T$ directly through $\bar{W} = (X_W,Y_W,Z_W)^T$, the center of the viewing window. The observer looks through the window with a telescope with magnification factor $MAG > 0$ and coefficient of optical transmission $B_o \in [0,1]$. The relationships between the observer, the viewing window, and the point of observation are pictured in Figure 9.

Specification of the observer defines his "cone of vision" into the world. Only those objects within this cone are potentially visible. One further specification is made: Only those objects that lie on the far side of the viewing window will be allowed to be visible. This reduces the observer's vision to the "pyramid of vision" (Figure 10).

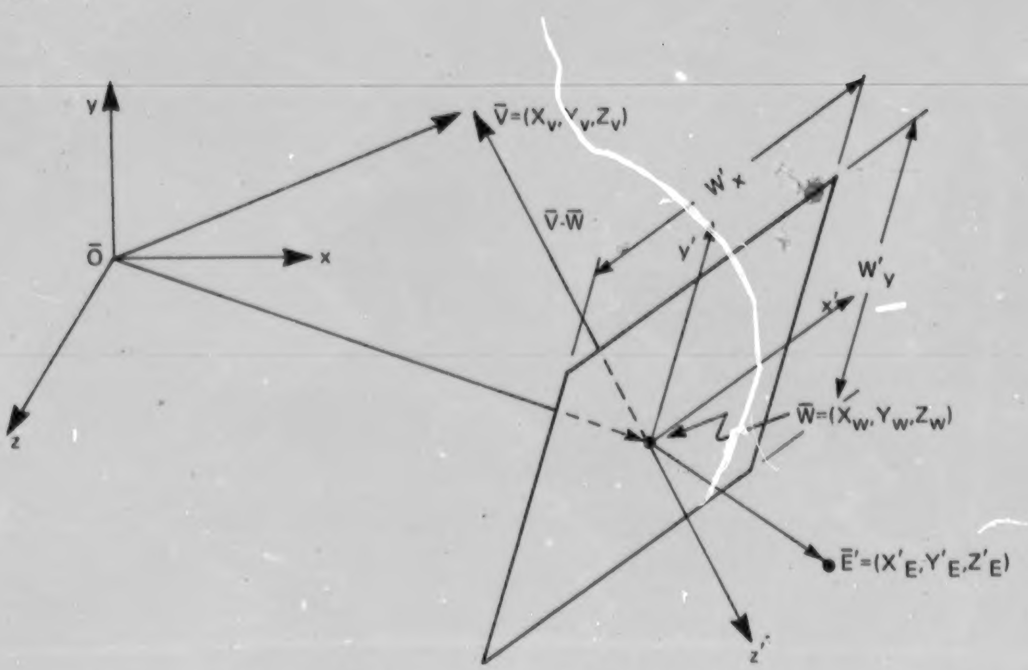


Figure 9 — Observer Viewing Orientation in R^3

World Description: Light Sources

Point light sources in R^3 are defined by the triplet $L_i = (\bar{x}_i, B_i, r_{o_i})$ where $\bar{x}_i$ represents the location of the i^{th} point light source, $B_i > 0$ represents its intensity, and r_{o_i} defines its brightness fall-off ratio. The source radiates its light evenly from its point location.

The sun light source is defined by the pair $(\bar{S}_x, B_s)$ where $\bar{S}$ represents the light direction and $B_s > 0$ represents the intensity. This sun light source provides parallel light rays throughout R^3.

World Description: Object Definition Requirements

The task of mathematically representing a complex object and all of its physical characteristics in R^3 may be quite formidable. This problem is simplified if this representation is limited to those parameters required to dynamically position the object in R^3 and to view it from any orientation in R^3.

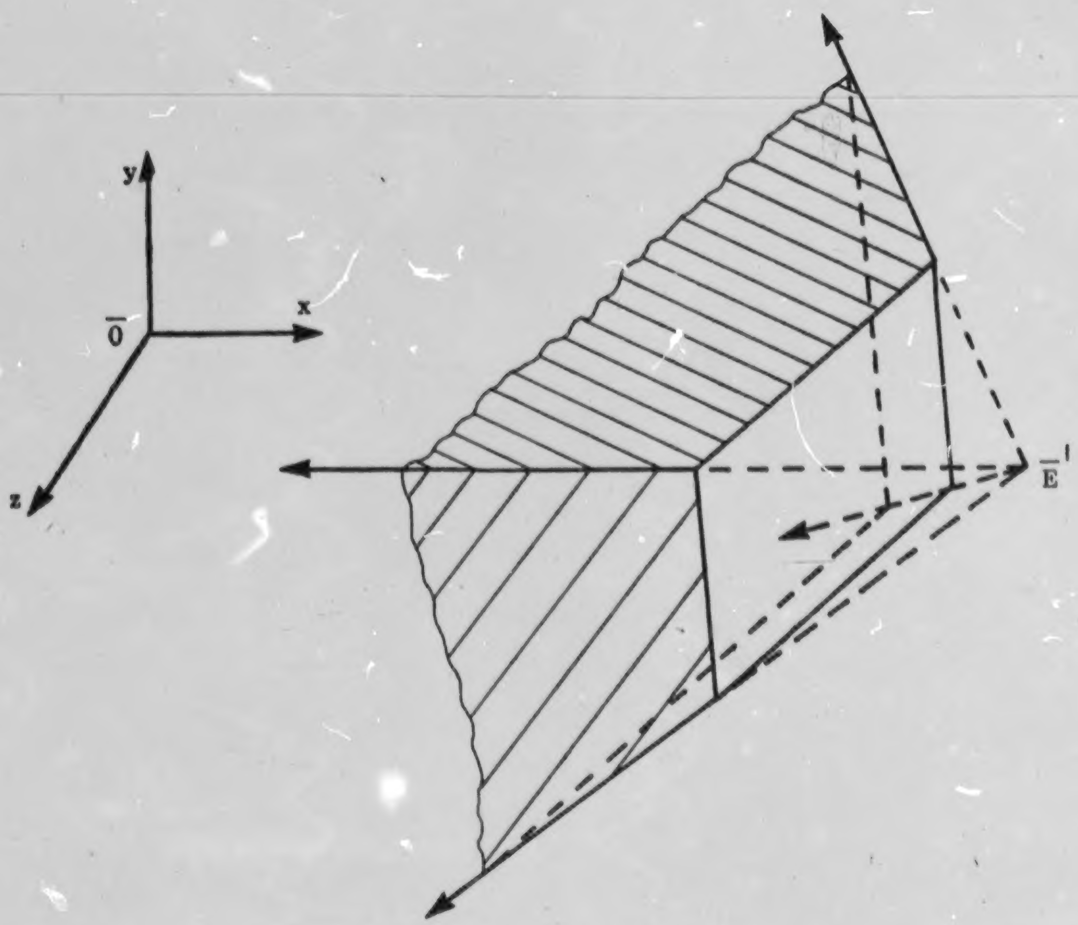


Figure 10— Pyramid of Vision

Consider an arbitrary object. The object is said to be convex only if the region of space S included within the object can be compressed as:

$$S = \left\{ (x, y, z) \mid f_i(x, y, z) \leq 0, i = 1, \ldots, n \right\} \qquad [1.1]$$

where $f_1, \ldots, f_n$ are real-valued functions such that for $i = 1, \ldots, n$:

$$f_j(x, y, z) \begin{cases} = 0, \text{ if } j = i \\ \leq 0, \text{ if } j \neq i \end{cases} \qquad [1.2]$$

define the object's n boundary conditions or surfaces. If an object is not convex, set S is the union of some m convex sets of the form of [1.1]. S is, in general, an infinite set regardless of whether or not it is a convex set.

If the object is to have a dynamical description in R^3, set S must be given a mass density function dM such that:

$$M = \int_S dm \tag{1.3}$$

where M is the total mass of set S. If the object is rigid, a point mass representing the total mass M of the set S and located at the center of mass $\overline{G} = (x_g, y_g, z_g)^T$ is sufficient. The center of mass is given by:

$$\begin{aligned} x_g M &= \int_S x\,dM \\ y_g M &= \int_S y\,dM \\ z_g M &= \int_S z\,dM \end{aligned} \tag{1.4}$$

If the object is homogeneous (i.e., dM = constant), the center of mass G coincides with the centroid $\overline{C} = (x_c, y_c, z_c)^T$ of set S given by:

$$\begin{aligned} V &= \int_S dV \\ x_c V &= \int_S x\,dV \\ y_c V &= \int_S y\,dV \\ z_c V &= \int_S z\,dV \end{aligned} \tag{1.5}$$

where dV = dxdydz is the differential volume element of R^3.

If an object is not itself rigid, but is made up of n rigid subobjects, then

$$\begin{aligned} M &= \sum_{i=1}^{m} M_i & V &= \sum_{i=1}^{m} V_i \\ x_g M &= \sum_{i=1}^{m} x_{g_i} M_i & x_c V &= \sum_{i=1}^{m} x_{c_i} V_i \\ y_g M &= \sum_{i=1}^{m} y_{g_i} M_i & y_c V &= \sum_{i=1}^{m} y_{c_i} V_i \\ z_g M &= \sum_{i=1}^{m} z_{g_i} M_i & z_c V &= \sum_{i=1}^{m} z_{c_i} V_i \end{aligned} \tag{1.6}$$

Since the only parts of an object which are potentially visible to an observer are its surfaces, the object at any instant of time may be adequately represented for display purposes by an appropriate description of its surfaces—the n boundary constraints of [1.1]—at that instant of time rather than the infinite set S itself. Color, opaqueness, and optical reflectivity are natural properties of surfaces and will be included within the surface definition.

It should be further noted that for homogeneous objects these n boundary constraints define the limits of integration of [1.3] and [1.4] and, hence, define the center of mass of the object and the mass of the object to within a multiplicative constant.

In summary, an object definition requires a defined mass, a center of mass, and a description of the object's surfaces.

World Description: Objects, Atoms, and Transformations

As explained in the previous section, an object's definition must include:

(1) A mass and a center of mass for dynamical representation in constant acceleration fields.

(2) A description of the object's surfaces for visual representation of the object at each instant of time.

Specification of these data can be quite involved, particularly if an object is complex. Furthermore, objects often contain similar basic shapes: cubes, parallelopipeds, ellipsoids, and so forth. Accordingly, various levels for object are provided.

The lowest level of object definition is the *atom.* An atom is defined to be that set of data that specifies (1) and (2) as above. As each atom is defined, it is tagged with an atom name. Atoms form the basic building blocks from which complex objects may be built.

An *object* is now defined to be an aggregate of *transformed* atoms and other objects.

A *transformation* is defined to be one of the following mathematical operations on atom data:

(1) $RC(C_g,C_b,C_r)$: Recolor each surface with the color[1] $C = C_g$ (green) + C_b (blue) + C_r (red).

(2) $C(C_g,C_b,C_r)$: Color any surface that has not already been colored with the color $C = C_g + C_b + C_r$.

---

[1] Here color is additive (e.g., red + green yields yellow).

(3) B(b): Multiply the coefficient of optical reflectivity of each surface by b.

(4) $S(k_x,k_y,k_z)$: Scale by k_x in x,k_y in y, and k_z in z.

(5) $M(m_x,m_y,m_z)$: Move by m_x in x, m_y in y, and m_z in z.

(6) $X(x,y,z,\theta)$: Rotate θ degrees about a line parallel to the x axis and passing through the point (x,y,z).

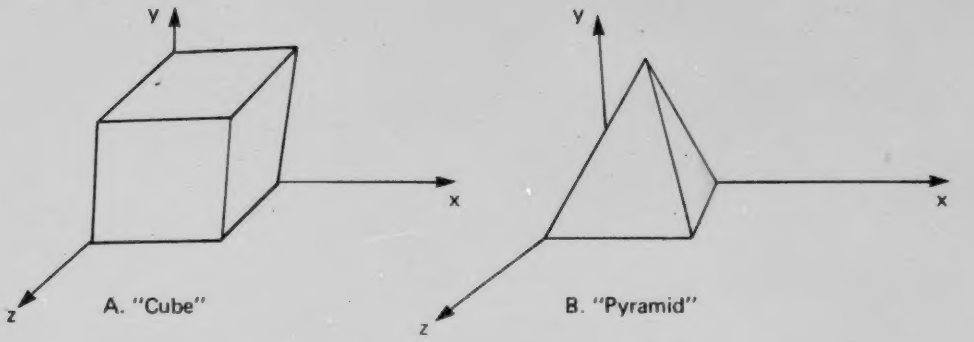
(7) $Y(x,y,z,\theta)$: Rotate θ degrees about a line parallel to the y axis and passing through the point (x,y,z).

(8) $Z(x,y,z,\theta)$: Rotate θ degrees about a line parallel to the z axis and passing through the point (x,y,z).

(9) W(w): Multiply the mass by w.

This scheme of definition allows a user to build increasingly complicated objects out of an existing set. At no time does the user have to be concerned with what the basic atom data are; he merely creates new objects out of existing ones. What the user does, in effect, is to create a "blueprint plan" for an object; then, upon receipt of a display command, the objects and subobjects are constructed from the atom data in accordance with the blueprint plan.

For example, suppose two atoms exist—"Cube" and "Pyramid"—and represent the basic shapes of Figure 11a and b. A user may then create the four-legged table of Figure 12 by creating a "Tabletop" out of a "Cube," a "Tableleg" out of a "Cube," and then "Table" out of "Tabletop" and four "Tableleg"s.



A. "Cube"

B. "Pyramid"

Figure 11 — Standard Orientation of Atoms in "Table"

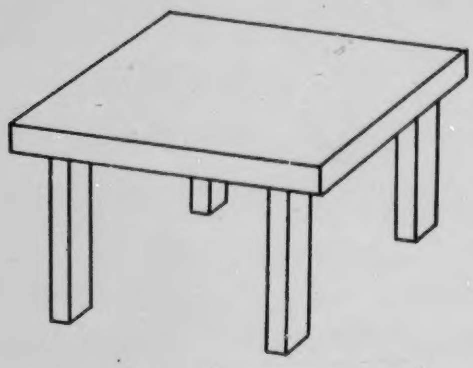
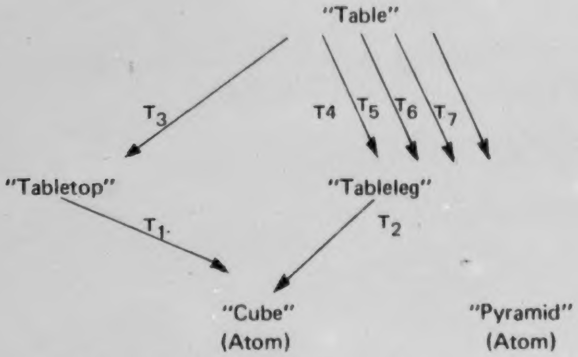


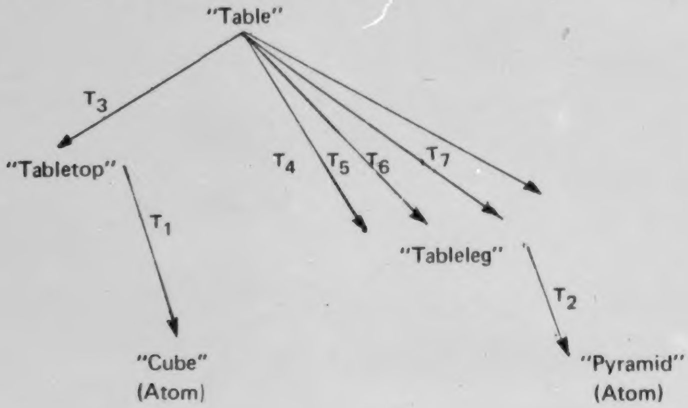
Figure 12– **"Table" (original specification)**

The blueprint structure specified by the user is:



It should be noted that an atom is always at the lowest level. Each T is a set of transformations and serves to link the various subobjects together.

At this stage, the user can modify any of three objects: "Table," "Tabletop," or Tableleg." The highest level of definition is two. Any change that the user makes in one level will ripple its effects into higher levels. For example, suppose the user redefines "Tableleg" to use "Pyramid" rather than "Cube." The data structure is now:



Display of "Table" now results in displaying an object with the shape of Figure 13. Each of the objects defining "Table" still appears, but "Tableleg" appears with its new definition.

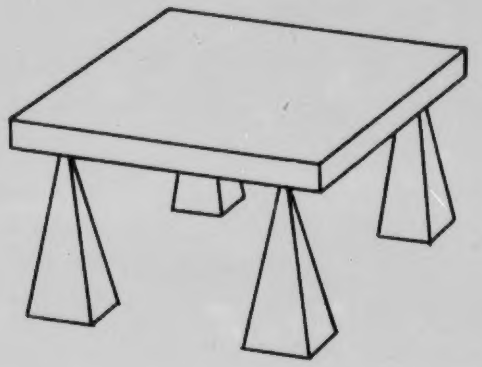


Figure 13 – "Table" ("Tableleg" modified)

It should further be noted that this scheme is open-ended as to the level of definition—"Table" may now be used as easily as "Tabletop," "Tableleg," "Cube," and "Pyramid." For example, the user could create a set of several "Table"s and call it "Tables."

The one thing that a user is not allowed to do is to attempt to form a recursive definition. Any attempt to modify "Tabletop" by adding "Table" to it would result in an error message since "Table" is defined, in part, by "Tabletop."

World Description: Atom Data (Definition)

Atoms form the data base from which all objects are constructed. The data for each atom contain the following information:

1. Mass and the location of the center of mass. (These data are used in the dynamics of objects.)
2. A description of the atom's surfaces. (These data define the visual properties of the atom: shape, location, and orientation in R^3, color, optical reflectivity, and so forth.)

Object transformations operate on and modify these data, but it is still transformed atom data that are decoded into a visual perspective by the image generator (except for color: $C(C_g,C_b,C_r)$ and $RC(C_g,C_b,C_r)$ are replacements).

The first requirement is satisfied by encoding a quadruplet (x_g,y_g,z_g,M) representing mass M situated at $\overline{G} = (x_g,y_g,z_g)^T$ in R^3.

The second requirement involves encoding the boundary constraints of the type of [1.1] and [1.2] to allow a projection processor to project the surfaces from R^3 into R^2 (i.e., to project the 3-dimensional world onto the 2-dimensional viewing window of the observer) in such a way as will then allow a visible surface processor to determine which of the projected surfaces is actually closest to the observer (i.e., which surfaces are visible and which surfaces are hidden behind closer surfaces).

This problem of projecting an object defined in 3 dimensions onto a 2-dimensional viewing window and representing its depth (the missing dimension) at any point on the viewing window is greatly simplified if the object contains only surfaces that lie on planes; that is, any point (x,y,z) on a surface satisfies an equation of the form

$$ax + by + cz + d = 0$$

where a,b,c, and d are scalar constants. Under this restriction it can be shown that $1/z$ is linearly related to (x',y'), the R^2 coordinates of the projected point (see projection processor). Hence, a general linear algorithm can be developed for solving the hidden surface problem if the original surfaces are planar.

Such planar-surface objects can easily be represented by finite sets of points, surfaces, and straight line edges in the x,y,z coordinate system. This is due to the fact that the n boundary constraints of [1.1] and [1.2] for the object reduce to a set of n linear equations if all of the surfaces are planar. For example, the unit volume cube of Figure 14 has the form

$$S = \left\{ (x,y,z) \mid x \geqslant 0, x+1 \geqslant 0, y \geqslant 0, y+1 \geqslant 0, z \geqslant 0, z+1 \geqslant 0 \right\}$$

and a point on the surface must satisfy at least one of the following:

$$\begin{array}{llll}
S_1: & (x=0, & 0 \leqslant y \leqslant 1, & 0 \leqslant z \leqslant 1) \\
S_2: & (x=1, & 0 \leqslant y \leqslant 1, & 0 \leqslant z \leqslant 1) \\
S_3: & (0 \leqslant x \leqslant 1, & y=0, & 0 \leqslant z \leqslant 1) \\
S_4: & (0 \leqslant x \leqslant 1, & y=1, & 0 \leqslant z \leqslant 1) \\
S_5: & (0 \leqslant x \leqslant 1, & 0 \leqslant y \leqslant 1, & z=0) \\
S_6: & (0 \leqslant x \leqslant 1, & 0 \leqslant y \leqslant 1, & z=1)
\end{array}$$

Each of the constraint surfaces is completely enclosed by a set of straight-line edges. Each straight-line edge is the locus of points that satisfy two boundary constraints. Each straight-line edge terminates at a vertex of the object—a point that satisfies three or more of the boundary constraints.

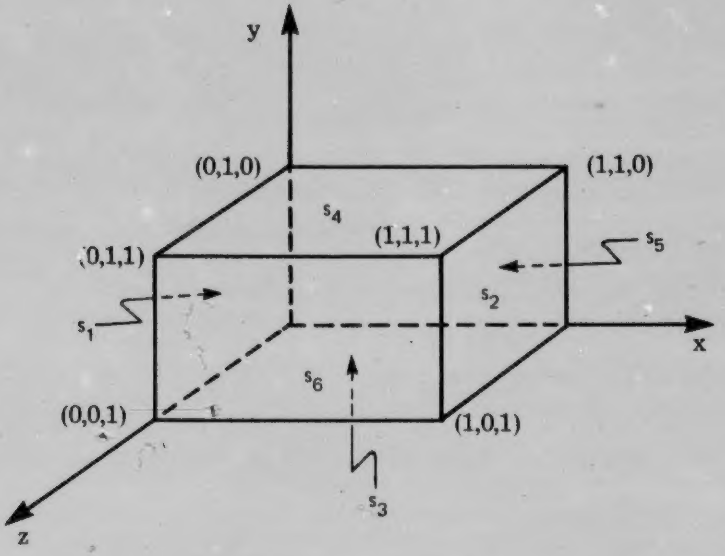


Figure 14 – **Unit Volume Cube**

Thus, objects with planar surfaces may be easily defined by overlaying them with a grid structure of points, surfaces, and straight-line edges from one object vertex point to another, separating one surface from another:

Points—Point data may be defined by either

(1) Direct input of an (x,y,z) coordinate.

(2) Specification of two of the three coordinate values and a surface. The third coordinate value is calculated by constraining the point to lie in the plane of the surface.

Surfaces—Surfaces are defined by the input of a triplet $(\overline{P}_1,\overline{P}_2,\overline{P}_3)$. The order of the triplet is such that the vector $\overline{N} = (\overline{P_1 - P_2}) \times (\overline{P_3 - P_2})$ points perpendicularly out of the surface. The surface is defined to be white $(C_g = C_b = C_r)$ and has coefficient of optical reflectivity equal to unity. Equivalent definitions for the plane surface S of Figure 15 are:

$$(\overline{P}_c,\overline{P}_b,\overline{P}_a) = (\overline{P}_b,\overline{P}_a,\overline{P}_c) = (\overline{P}_a,\overline{P}_c,\overline{P}_b)$$

Edges—Edges are defined by the input of a quadruplet $(\overline{P}_1,\overline{P}_2,S_L,S_R)$, such that if the edge were transversed from point $\overline{P}_1$ to point $\overline{P}_2$, S_L is the surface on the left side of the edge and S_R is the surface on the right side of the edge. See Figure 16.

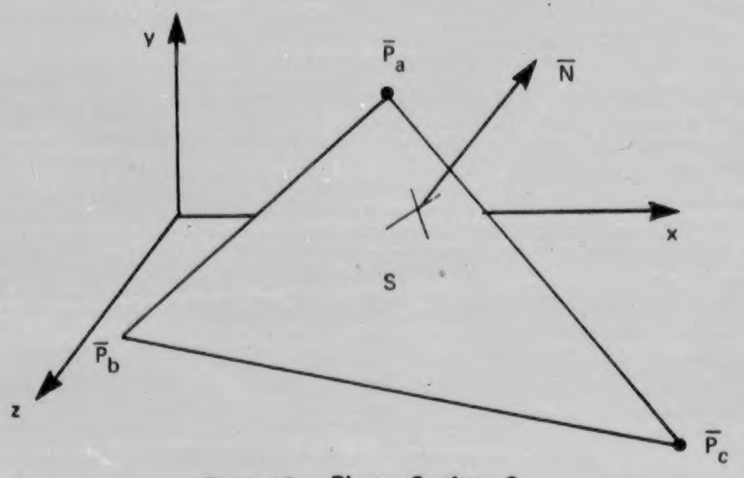


Figure 15 — **Planar Surface S**

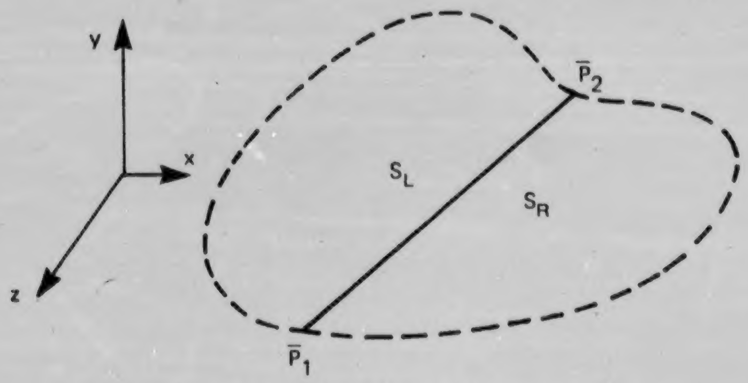


Figure 16— **Edge Between Planar Surfaces S_L and S_R**

Non-planar analytic surfaces may be handled with a minor addition to the grid structure. Consider the smooth (i.e., analytic) surface S_{sm} of Figure 17. The surface satisfies

$$f(\overline{x}) = f(x,y,z) = 0 \qquad [1.7]$$

and has continuous first partial derivatives. Since f is a scalar function, the gradient vector is given by

$$\overline{N}(\overline{x}) = \frac{\partial f(\overline{x})}{\partial \overline{x}} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} \qquad [1.8]$$

and points perpendicularly out of the surface at each point $\overline{x}$. Let $\overline{x}_0, \overline{x}_1$, and $\overline{x}_2$ be points on S_{sm} and let $\overline{N}(\overline{x}_0)$, $\overline{N}(\overline{x}_1)$, and $\overline{N}(\overline{x}_2)$ be the gradient at those points. Let S_{f1} be a planar surface passing through the points $\overline{x}_0, \overline{x}_1$, and $\overline{x}_2$.

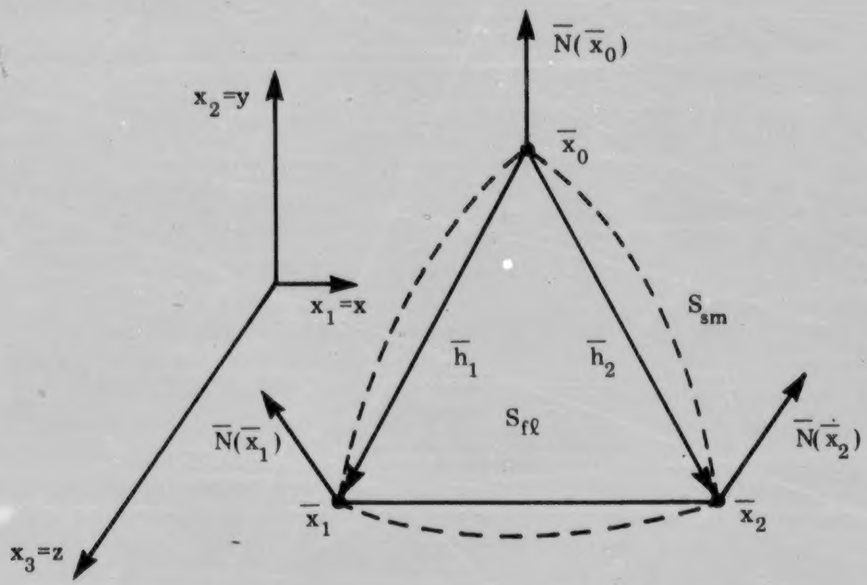


Figure 17 – **Smooth Surface S_{sm} and Flat Surface $S_{f\ell}$**

Since the surface is analytic in the region of interest, it may be represented by a Taylor's series about the point $\overline{x}_0$.

$$f(\overline{x}_0+\overline{h}) = f(\overline{x}_0) + f'(\overline{x}_0)(\overline{x}_0+h) + \frac{1}{2!}(\overline{x}_0+\overline{h})^T f''(\overline{x}_0)(\overline{x}_0+\overline{h}) + \ldots \quad [1.9]$$

$$= f(\overline{x}_0) + f'(\overline{x}_0)(\overline{x}_0+\overline{h}) + o(|\overline{x}_0+\overline{h}|)$$

where $f'(\overline{x}_0) = \overline{N}(\overline{x}_0)$.

Now let $\overline{x}_1 = \overline{x}_0 + k_1\overline{h}_1$ and $\overline{x}_2 = \overline{x}_0 + k_2\overline{h}_2$. Clearly, as $k_1, k_2 \rightarrow 0, \overline{x}_1, \overline{x}_2, \rightarrow \overline{x}_0$ and $\overline{N}(\overline{x}_1), \overline{N}(\overline{x}_2) \rightarrow \overline{N}(\overline{x}_0)$. Hence, the locus of points on a plane passing through the points $\overline{x}_0, \overline{x}_1$, and $\overline{x}_2$, given by

$$(\overline{x - x_0})^T \cdot [(\overline{x_1 - x_0}) \times (\overline{x_2 - x_0})] = 0 \quad [1.10]$$

approaches the points on the plane

$$(\overline{x - x_0})^T \cdot N(\overline{x}_0) = 0$$

formed by the first order terms of the Taylor's series expansion. The error is $o(|\overline{x - x_0}|)$. Thus, the analytic surface may be arbitrarily closely approximated by a set of planar surfaces.

All surfaces are detected by the observer as a color-brightness pair. The surface color is the color assigned to the object by a color transformation. The surface brightness at point $\overline{x}$ on the non-shiny surface is given by

$$B(x) = \mathrm{BFACT}\left[B_{AMB} + \sum_{i=1}^{p} B_i f\left(\frac{\overline{(\ell_i - x)}^T \cdot \overline{N}(\overline{x})}{\overline{\|\ell_i - x\|}\ \ \|\overline{N}(\overline{x})\|}\right) + B_S f\left(\frac{\overline{x}_s^{\,T} \cdot \overline{N}(\overline{x})}{\|\overline{x}_s\|\ \ \|N(\overline{x})\|}\right)\right] \tag{1.11}$$

where:

BFACT = coefficient of optical reflectivity of the surface
B_{AMB} = brightness of ambiant light source
P = number of point light sources
$\overline{\ell}_i$ = location of the i^{th} light source in R^3
B_i = brightness of the i^{th} light source
$\overline{N}(\overline{x})$ = normal to the surface at point $\overline{x}$
$\overline{S}$ = directional vector of light from sun at ∞
B_S = brightness of sun
$f(\alpha)$ = brightness reflectivity function

For planar surfaces $\overline{N}(x) = \overline{N}$, a constant vector, and [1.11] reduces to

$$B(\overline{x}) = \mathrm{BFACT}\left[B_{AMB} + \sum_{i=1}^{P} B_i f\left(-\frac{\overline{(\ell_i - x)}^T \cdot N}{\overline{\|\ell_i - x\|}\ \|N\|}\right) + B_S f\left(\frac{\overline{S}^T \cdot \overline{N}}{\|\overline{S}\|\ \|N\|}\right)\right] \tag{1.12}$$

If there are no point light sources, the brightness on a flat surface is constant and is given by

$$B(\overline{x}) = B = \mathrm{BFACT}\left[B_{AMB} + B_S f\left(\frac{\overline{S}^T \cdot \overline{N}}{\|S\|\ \|\overline{N}\|}\right)\right] \tag{1.13}$$

Now consider a non-planar analytical surface $f(\overline{x})$ arbitrarily closely approximated by a set of planar surfaces. Since $f(\overline{x})$ is analytic, $f'(\overline{x}) = \partial f(x)/\partial x = \overline{N}(\overline{x})$ is continuous. Therefore $B(\overline{x})$ as given by [1.11] is continuous; in particular, $B(\overline{x})$ is continuous across the edge separating one of the approximating planar surfaces from another. Since neither [1.12] nor [1.13] is continuous across these boundaries, a continuous brightness function must be defined over the grid structure.

Consider Figure 17 again. The surface S_{sm} corresponding to $f(\bar{x})$ is approximated continuously over R^3 by a set of planar surfaces that pass through a set of points $\{\bar{x}_i, f(\bar{x}_i) = 0\}$ known to lie on S_{sm}. If the gradient vector $\bar{N}(\bar{x}) = \partial f(\bar{x})/\partial \bar{x}$ is known at these points, then the brightness function $B(\bar{x})$ is known at that set of points from [1.11]. This known set of brightness $\{\bar{x}_i, B(\bar{x}_i)\}$ allows the definition of a continuous approximating brightness function.

Let $\lambda \in [0,1]$. Clearly, $\bar{x} = \lambda \bar{x}_i + (1 - \lambda)\bar{x}_j$ is a point that lies on the edge between $\bar{x}_i$ and $\bar{x}_j$ in the grid structure. Two obvious functions that could be used to approximate the brightness function $B(\bar{x})$ along this edge are:

$$\text{Linear Rule:} \quad \hat{B}(\lambda) = \lambda B(\bar{x}_i) + (1-\lambda) B(\bar{x}_j), \ \lambda \epsilon [0,1] \qquad [1.14]$$

$$\text{Logarithmic Rule:} \quad \hat{\hat{B}}(\lambda) = B(\bar{x}_j) \left(\frac{B(\bar{x}_i)}{B(\bar{x}_j)} \right)^{\lambda}, \ \lambda \epsilon [0,1] \qquad [1.15]$$

For maximum dynamic range for a fixed number of bits, [1.15] was chosen. Rule [1.15] specifies the brightness around the perimeter of each of the approximating planar surfaces. The brightness at any point $\bar{x}$ within a perimeter will be defined as the result of applying [1.15] to an edge resulting from the intersection of the approximating planar surface with the $y = x_2$ plane passing through the point $\bar{x} = (x = x_1, y = x_2, z = x_3)$. See Figure 18.

Appendix A provides an example of a non-planar surface.

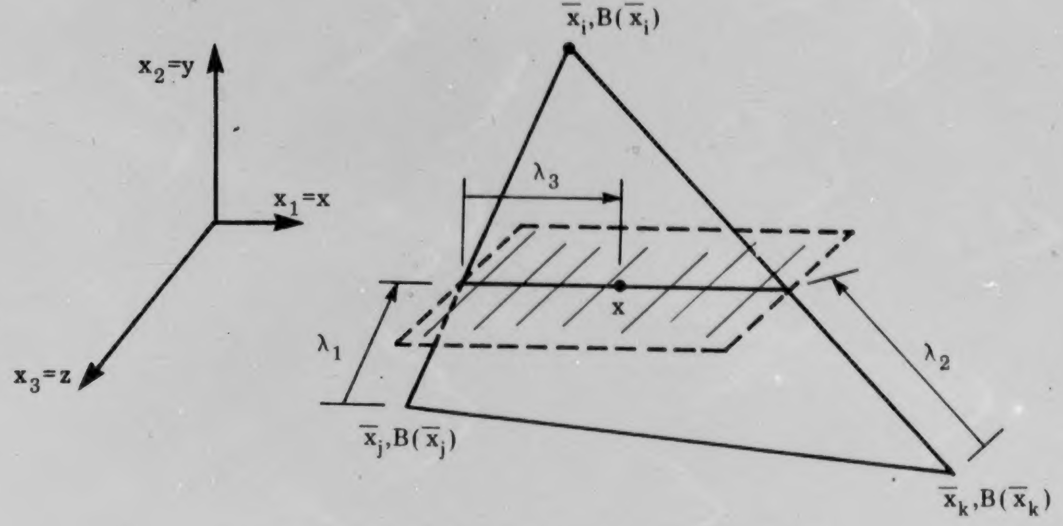

$\bar{x}_i, B(\bar{x}_i)$
$x_2 = y$
$x_1 = x$
$x_3 = z$
λ_3
λ_1
λ_2
x
$\bar{x}_j, B(\bar{x}_j)$
$\bar{x}_k, B(\bar{x}_k)$

Figure 18 — **Brightness Within Planar Surface Perimeter**

Final Atom Data

Once the grid structure over an atom has been completed, the atom is preprocessed and put into the atom library in a compact form. This preprocessing involves:

Points: All point values are calculated and points are stored as an $\overline{x} = (x,y,z)^T$ 96-bit floating point triplet.

Surfaces: All surface normals are calculated to insure that the $(\overline{P}_1,\overline{P}_2,\overline{P}_3)$ triplet does, indeed, define a surface normal $\overline{N} = (N_x,N_y,N_z)^T$.

Brightness Normals: Brightness normals are calculated for vertex points on analytic, non-planar surfaces. These normals are held to 10-bit precision per component.

Edges: The general form of an edge becomes

$$E = (P_1,P_2,P_3,P_4,S_L,S_R,B_1,B_2,B_3,B_4)$$

where

P_1 = label of end point $\overline{P}_1$
P_2 = label of end point $\overline{P}_2$
P_3 = label of point on left surface
P_4 = label of point on right surface

S_L = label of left surface going from $\overline{P}_1$ to $\overline{P}_2$
S_R = label of right surface going from $\overline{P}_1$ to $\overline{P}_2$

B_1,B_2,B_3,B_4 appear only if S_L or S_R is a non-planar surface. If S_L and S_R are approximating planes for the same analytic surface, B_1 is the label of the brightness normal at $\overline{P}_1$, and B_2 is the brightness normal at $\overline{P}_2$. Otherwise, B_1 and B_2 refer to brightness normals for S_L at $\overline{P}_1$ and $\overline{P}_2$, and B_3 and B_4 refer to brightness normals for S_R at $\overline{P}_1$ and $\overline{P}_2$.

The atom data remembered in the atom library and used in decoding the image are:

(1) Point data
(2) Brightness normal data
(3) Edge data

Section 2

PROJECTION PROCESSOR

INTRODUCTION

This section provides a detailed description of the projection processor of the CHARGE image generator. The projection processor accepts as input a description of a 3-dimensional space from an observer within this space. The nature of this description is detailed in Section 1, World Description and Viewing Parameters (page 20). The projection processor performs two functions:

(1) It reduces the total set of 3-dimensional world description data into a subset that is potentially visible to the observer.

(2) It transforms this set of potentially visible 3-dimensional data into an equivalent set of encoded 2-dimensional potentially visible data.

The latter data set serves as input to the high-speed linear visible surface processor that forms the video image and then encodes it for display on a CHARGE display terminal.

Input Data

Input data to the projection processor contains the following information:

(1) Specification of the observer in R^3

(2) Specification of the light sources in R^3

(3) Mappings of atoms into R^3

Observer

The observer is completely specified by the set of parameters $(X_W,Y_W,Z_W,X'_E,Y'_E,Z'_E,X_V,Y_V,Z_V,W'_X,W'_Y,MAG,B_o)$ which define an eye, a viewing window, and a point of observation. These parameters are pictorially represented in Figure 19 $\overline{V} = X_V,Y_V,Z_V)^T$ defines the point of observation with respect to $\overline{0} = (0,0,0)^T$, the origin of R^3. This is the point toward which the observer directs his vision. The observer sees into R^3 only through a viewing window. This viewing window has a center located at $\overline{W} = (X_W,Y_W,Z_W)^T$ with respect to the origin of R^3, and lies in a plane orthogonal to the vector $\overline{V} - \overline{W}$.

$\overline{W}$ is the origin of a second rectangular axis coordinate system with axes x', y', z' and unit directional vectors $\overline{i}', \overline{j}'$, and $\overline{k}'$ defined so that:

$$\begin{array}{lll} \overline{i}' \times \overline{i}' = \overline{0} & \overline{j}' \times \overline{i}' = -\overline{k}' & \overline{k}' \times \overline{i}' = \overline{j}' \\ \overline{i}' \times \overline{j}' = \overline{k}' & \overline{j}' \times \overline{j}' = \overline{0} & \overline{k}' \times \overline{j}' = -\overline{i}' \\ \overline{i}' \times \overline{k}' = -\overline{j}' & \overline{j}' \times \overline{k}' = \overline{i}' & \overline{k}' \times \overline{k}' = \overline{0} \end{array}$$

$$\overline{k}' \cdot \frac{(\overline{V}' - \overline{W})}{||\overline{V}' - \overline{W}||} = -1$$

$$\overline{i}' \cdot \overline{j} = \overline{0}$$

With respect to this primed coordinate system, the window size is W'_X horizontally by W'_Y vertically, and the y and y' axes lie in a plane. The observer's eye is located at $\overline{E}' = (X'_E, Y'_E, Z'_E)^T$ with respect to the primed axes. If $X'_E = Y'_E = 0$, the eye looks at $\overline{V} = (X_V, Y_V, Z_V)^T$ directly through $\overline{W} = (X_W, Y_W, Z_W)^T$, the center of the viewing window. The observer looks through the window with a telescope with magnification factor $MAG > 0$ and coefficient of optical transmission $B_o \in [0,1]$. The relationships between the observer, the viewing window, and the point of observation are pictured in Figure 19.

Light Sources

The types of light sources are provided for: a planar light source and point light sources. The planar light source is specified by defining a sun vector and brightness level fraction

$$L_s = (\overline{x}_s, B_s) \qquad [2.1]$$

where $\overline{x}_s = (x_s, y_s, z_s)^T$ specifies the direction of the parallel light rays through R^3, and $B_s \in [0,1]$ specifies the brightness as a fraction of B_{MAX}, the maximum allowed brightness. This sunlight source illuminates a "non-shiny" surface with gradient vector $\overline{N}(\overline{x}) = \frac{\partial f(\overline{x})}{\partial \overline{x}}$ with brightness

$$B(x) = f\left(\frac{\overline{x}_s{}^T \overline{N}(\overline{x})}{||\overline{x}_s|| \;\; ||\overline{N}(\overline{x})||}\right) B_s B_{MAX} \qquad [2.2]$$

where $f(\alpha)$ is a reflectivity function. All surfaces with the same gradient vector will be illuminated with the same brightness.

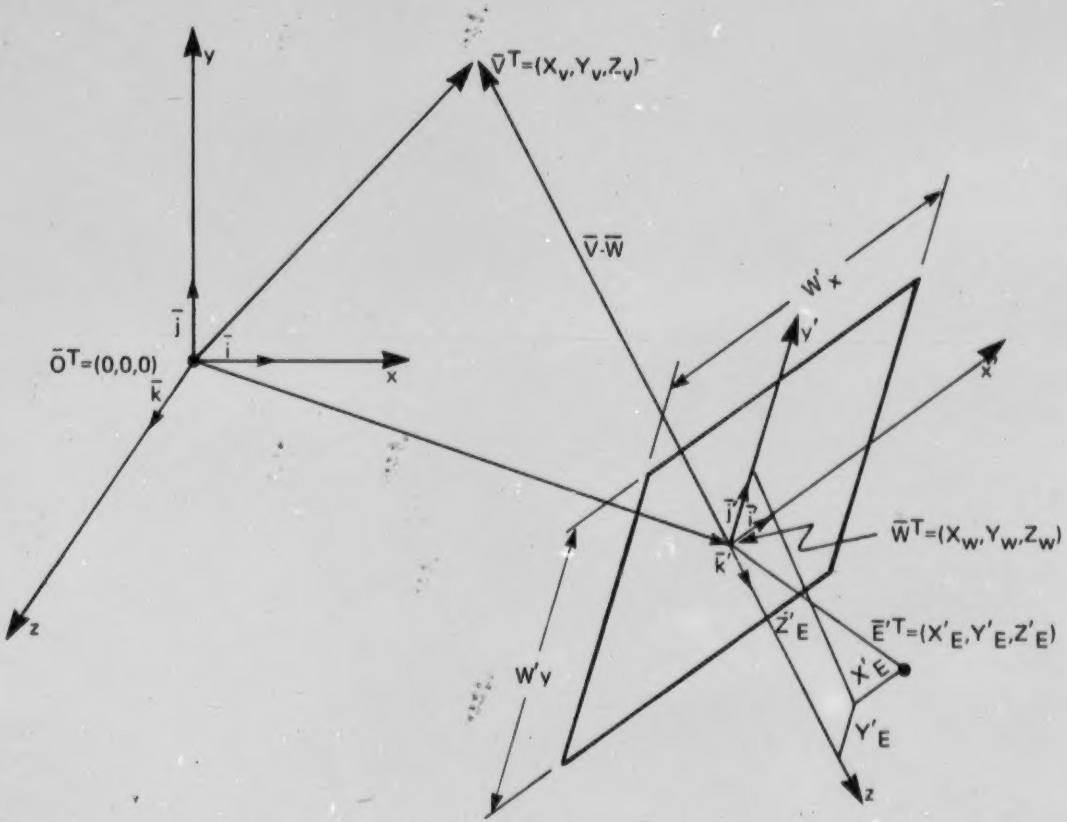


Figure 19 — **Observer in R^3**

A point light source is specified by defining a location vector, a brightness level fraction, and a brightness fall-off fraction.

$$L_i = (\bar{x}_i, B_i, r_i) \qquad [2.3]$$

where $\bar{x}_i = (x_i, y_i, z_i)^T$ specifies the location of the i^{th} point light source of brightness $B_i B_{MAX}$ ($B_i \in [0,1]$) at $\bar{x}_i$. The point light source emits light radially from $\bar{x}_i$ with brightness diminishing inversely with the square of the normalized radical distance $({}^r/r_{o_i})$ from x_i

$$|B(r/r_{o_i})| = \left[\frac{1}{({}^r/r_{o_i})^2 + 1}\right] B_i B_{MAX} \qquad [2.4]$$

This point light source illuminates a surface at point $\overline{x}$ with gradient vector $(\overline{N}(\overline{x}) = \partial f(\overline{x})/\partial \overline{x}$ with brightness

$$B(\overline{x}) = f\left(\frac{(\overline{x}_i - \overline{x})^T \,\overline{N}(\overline{x})}{\|\overline{x}_i - \overline{x}\| \;\; \|\overline{N}(\overline{x})\|}\right) \frac{B_i B_{MAX}}{\left(\frac{\|\overline{x}_i - \overline{x}\|}{r_{o_i}}\right)^2 + 1} \qquad [2.5]$$

as shown in Figure 20. The world description contains only one sunlight source, but may contain an arbitrary number of point light sources.

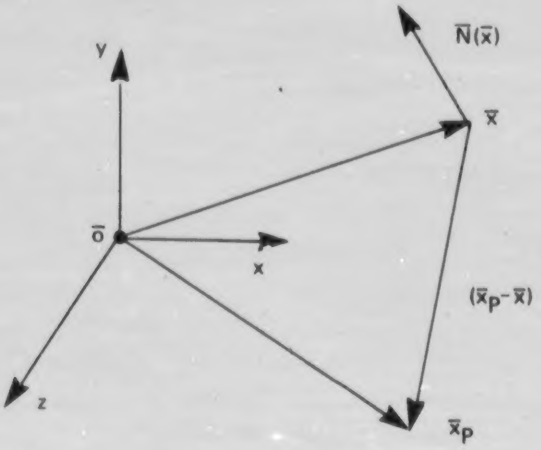

$\overline{N}(\overline{x})$
y
$\overline{x}$
$\overline{o}$
x
$(\overline{x}_p - \overline{x})$
z
$\overline{x}_p$

Figure 20 — **Brightness at Point $\overline{X}$ Due to Point Source at Point $\overline{X}_P$**

Atoms

Atoms, the basic building blocks of objects in the world, are mapped into their final orientations in R^3 by composite transformations that link object definitions from their highest level down to the atom level. Transformations that map the atom's point data in R^3 are implemented via 4 x 4 matrix operators. These are:

$$M(m_x, m_y, m_z) = \begin{bmatrix} 1 & 0 & 0 & m_x \\ 0 & 1 & 0 & m_y \\ 0 & 0 & 1 & m_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.6]$$

$$S(k_x, k_y, k_z) = \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.7]$$

$$X(x, y, z, \theta_x) = M(x, y, z) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} M(-x, -y, -z) \quad [2.8]$$

$$Y(x, y, z, \theta_y) = M(x, y, z) \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} M(-x, -y, -z) \quad [2.9]$$

$$Z(x, y, z, \theta_z) = M(x, y, z) \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} M(-x, -y, -z) \quad [2.10]$$

The composite transformation [T*] that maps an atom in an object of level of definition m into its position in R^3 is given by

$$[T^*] = [T_m][T_{m-1}] \ldots [T_3][T_2][T_1] \quad [2.11]$$

where $[T_k]$ is the transformation (which may itself be composite) which defines the atom's orientation in the k^{th} level of object definition in terms of its orientation in the $(k-1)^{th}$ level of object definition.

[T*] maps a point $\overline{P}_o = (x_o, y_o, z_o, 1)^T$ of the original atom definition into its final position $\overline{P}_f = (x_f, y_f, z_f, 1)^T$ via the matrix product

$$\overline{P}_f = [T^*]\overline{P}_o \quad [2.12]$$

The atom definition is such that its final form in R^3 is strictly a function of its points in R^3. Hence, [T*]: Atom (original) → Atom (final orientation in R^3).

Processing Steps

The projection processor projects the 3-dimensional world description

$$\left(\text{Observer, Sunlight Source, } \{\text{Pt. Light Sources}\},\ \{([T_i],\ \text{Atom}_i)\}\right)$$

into an encoded 2-dimensional description via the processing steps of Figure 21.

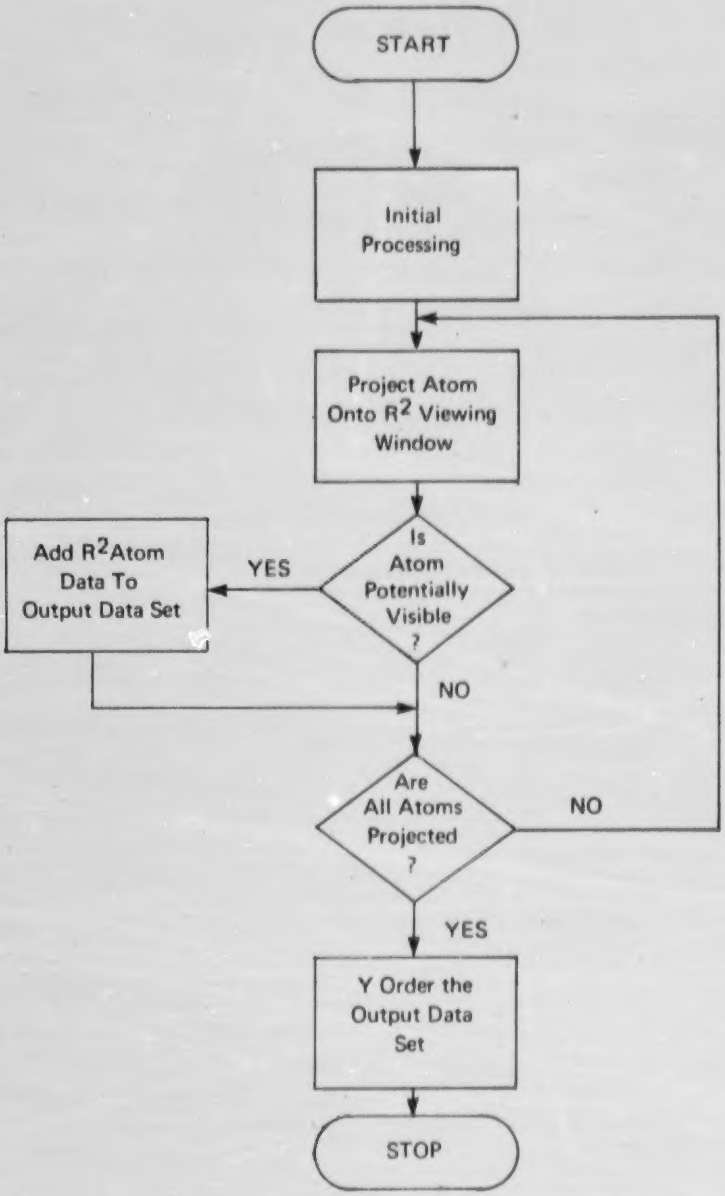

START
Initial Processing
Project Atom Onto R2 Viewing Window
Is Atom Potentially Visible ?
YES
Add R2Atom Data To Output Data Set
NO
Are All Atoms Projected ?
NO
YES
Y Order the Output Data Set
STOP

Figure 21 — **Projection Processor Processing Steps**

The amount of processing is roughly equal to the number of atoms in the R^3 world description times the average atom complexity. The output data set that will be passed to the visible surface processor may contain data from all of the atoms or, at the other extreme, it may be empty.

Initial Processing

The job of the projection processor is to encode the R^3 world description onto the R^3 viewing window in accordance with the observer's parameters. This is most easily accomplished if (a) the observer's eye is located at the origin $\overline{0}$ of R^3; (b) the observer's viewing window is located in a constant z plane with the x′ axis parallel to the x axis, the y′ axis parallel to the y axis, and the z′ axis parallel to the z axis; and (c) the observer's apparent magnification factor is unity. Let $[T_w]$ denote the matrix transformation required to do this. If $[T_w]$ is applied to the observer, the observer's viewing window, each of the light sources, and each of the atoms in R^3, then the relative positioning of this transformed input data set is equivalent to that of the original input data set.

Transformation $[T_w]$ may be computed in two stages from the observer's parameters. The matrix transformation

$$[T'_w] = [T_{MAG}]\,[T_x]\,[T_y]\,[T_{m1}]$$

where

$$[T_{m1}] \; : \; M(-X_w, -Y_w, -Z_w)$$

$$[T_y] \; : \; Y(0,0,0,\theta_y) \qquad \theta_y = \tan^{-1} \frac{(X_v - X_w)}{\sqrt{(Z_v - Z_w)^2}}$$

$$[T_x] \; : \; X(0,0,0,\theta_x) \qquad \theta_x = \tan^{-1} \frac{-(Y_v - Y_w)}{\sqrt{(X_v - X_w)^2 + (Z_v - Z_w)^2}}$$

$$[T_{MAG}] : \; S(MAG, MAG, MAG)$$

moves the observer and his viewing window from the orientation of Figure 19 into that of Figure 22. The primed and unprimed axes now are coincident with the center of the window now lying at $\overline{0}$, the origin of R^3. The viewing window now lies in the $z = 0$ plane and the eye is located at $\overline{E} = (X_E, Y_E, Z_E)^T$ where $X_E = MAG \cdot X'_E$, $Y_E = MAG \cdot Y'_E$, and $Z_E = MAG \cdot Z'_E$. If the viewing window is left the same W'_x by W'_y size, the desired magnification is accomplished.

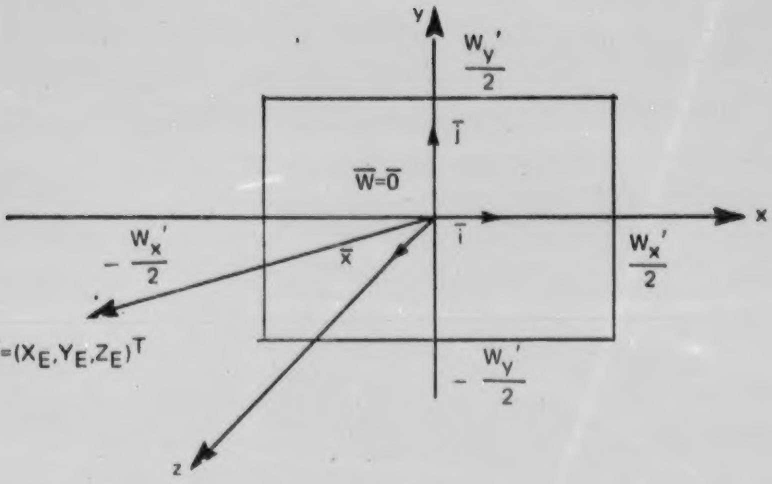


Figure 22 – **World After $[T'_w]$ Transformation**

The final desired magnification may be obtained by applying a move transformation of $M(-X_E,-Y_E,-Z_E)$. This transformation applies to the window coordinates as well this time and results in the observer-viewing window orientation of Figure 23.

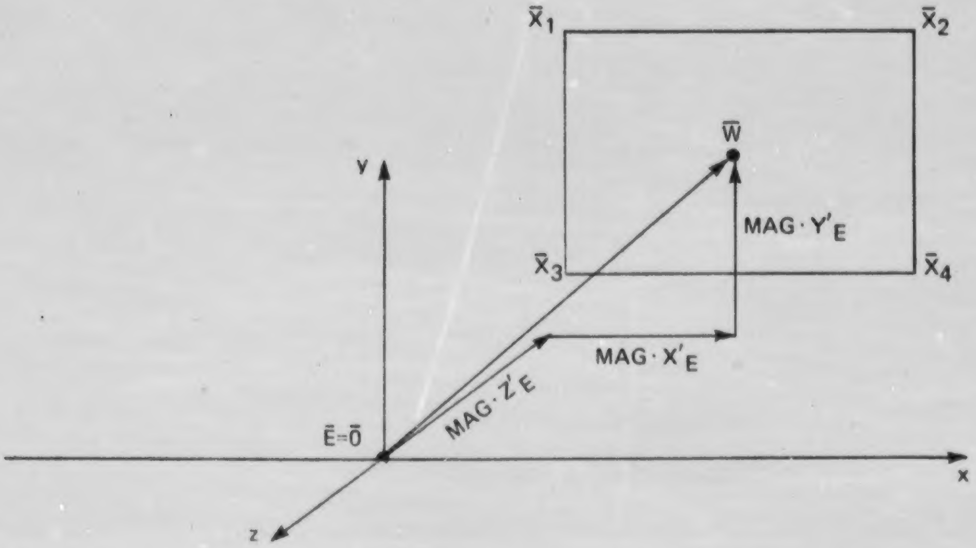


Figure 23 – **World After $[T_W]$ Transformation**

And so

$$[T_W] = [T_{M2}]\,[T_{MA1}]\,[T_X]\,[T_Y]\,[T_{M1}] \qquad [2.13]$$

where

$[T_{M1}]$: $M(-X_w, -Y_w, -Z_w)$

$[T_Y]$: $Y(0,0,0,\theta_y)$ $\qquad \theta_y = \tan^{-1} \dfrac{(X_v - X_w)}{\sqrt{(Z_v - Z_w)^2}}$

$[T_X]$: $X(0,0,0,\theta_x)$ $\qquad \theta_x = \tan^{-1} \dfrac{-(Y_v - Y_w)}{\sqrt{(X_v - X_w)^2 + (Z_v - Z_w)^2}}$

$[T_{MAG}]$: $S(MAG, MAG, MAG)$

$[T_{M2}]$: $M(-MAG \cdot X'_E, -MAG \cdot Y'_E, -MAG \cdot Z'_E)$

The remaining window now has corner coordinates:

$$\begin{aligned}
\overline{X}_1 &= \left(-\frac{W'_x}{2} - MAG \cdot X'_E, +\frac{W'_y}{2} - MAG \cdot Y'_E, -MAG \cdot Z'_E\right) = (X'_{MIN}, Y'_{MAX}, Z_{vw}) \\
\overline{X}_2 &= \left(+\frac{W'_x}{2} - MAG \cdot X'_E, +\frac{W'_y}{2} - MAG \cdot Y'_E, -MAG \cdot Z'_E\right) = (X'_{MAX}, Y'_{MAX}, Z_{vw}) \qquad [2.14] \\
\overline{X}_3 &= \left(-\frac{W'_x}{2} - MAG \cdot X'_E, -\frac{W'_y}{2} - MAG \cdot Y'_E, -MAG \cdot Z'_E\right) = (X'_{MIN}, Y'_{MIN}, Z_{vw}) \\
\overline{X}_4 &= \left(+\frac{W'_x}{2} - MAG \cdot X'_E, -\frac{W'_y}{2} - MAG \cdot Y'_E, -MAG \cdot Z''_E\right) = (X'_{MIN}, Y'_{MIN}, Z_{vw})
\end{aligned}$$

and the initial processing steps are:

$$\begin{aligned}
&(1) \quad (X'_E, Y'_E, Z'_E),\ (X_w, Y_w, Z_w),\ (X_v, Y_v, Z_v),\ MAG \rightarrow [T_w] \\
&(2) \quad L_s = (\overline{X}_s, B_s) \xrightarrow{[T_w]} \left(\frac{[T_w]\,\overline{X}_s}{\|[T_w]\,\overline{X}_s\|}, B_s\right) = (\overline{S}_n, B_s) \qquad [2.14a] \\
&(3) \quad L_i = (\overline{X}_i, B_i, r_{o_i}) \xrightarrow{[T_w]} ([T_w]\,\overline{X}_i, B_i, MAG \cdot r_i) = (\overline{\ell}_i, B_i, r'_{o_i})
\end{aligned}$$

As was mentioned, $[T_W]$ must also be applied to $([T_i], ATOM_i)$ pairs to maintain the original relative positioning of the viewer, light sources, and objects. Functionally

$$\text{Atom in atom library} \xrightarrow{[T_i]} \text{Atom in world} \xrightarrow{[T_w]} \text{Atom in transformed world}$$

This latter operation of transforming $([T_i], ATOM_i)$ pairs into $([T_w]\,[T_i], ATOM_i)$ pairs will be done in the atom projection loop.

Atom Projection Loop

Object data entering the projection processor consists of ($[T_i]$,ATOM$_i$) pairs where $[T_i]$ is a matrix transformation that maps the points of atom ATOM$_i$ into (original) R^3 position. Also available is matrix transformation $[T_w]$ that maps the original R^3 into an equivalent 3-dimensional space in which the center of the viewing window is at the origin of this new space. Atom data consists of (a) a finite set of points defining the atom's vertices in its standard (library) orientations, (b) a finite set of brightness normals representing the gradient vector $\partial f_i(\overline{x})/\partial\overline{x}$ of the atom's surfaces in the atom's standard (library) orientation, and (c) a finite set of straight line atom edges which link together the atom's surface constraints and brightness functions; the edge definition is invariant to the atom's orientation.

The first processing step is to transform the atom into its final orientation. Points are transformed via [2.15].

$$\overline{P}_f = [T_w]\,[T_i]\,\overline{P}_o \qquad [2.15]$$

where $[T_w][T_i]$ is the composite transformation mapping the point $\overline{P}_o{}^T = (x_o,y_o,z_o,1)$ into its final position $\overline{P}_f{}^T = (x_f,y_f,z_f,1)$. The set of surface normals $\{\partial f_i(\overline{x})/\partial\overline{x}\}$ are transformed via [2.16].

$$\overline{N}_f = \mathrm{Adj}^T\,\{[T_w]\,[T_i]\}\,\overline{N}_o \qquad [2.16]$$

where $\overline{N}_o$ is an initial gradient vector $\overline{N}_o{}^T = (n_x,n_y,n_z)$. [2.16] may be easily derived as follows: Let $\overline{N}$ be orthogonal to surface S and let $\overline{x}_o$ be a point on S.

Then $$\overline{N}^T \cdot (\overline{x} - \overline{x}_o) = 0 \qquad \forall\,\overline{x}\epsilon S,\ \overline{x} \neq \overline{x}_o$$

Now let $[T]$ be a composite matrix transformation mapping R^3 into R^3. Let $[T^*]$ be another matrix transformation such that

$$([T^*]\overline{N})^T \cdot \Big([T]\,(\overline{x} - x_o)\Big) = 0 \qquad \forall\,\overline{x}\epsilon S,\ \overline{x} \neq \overline{x}_o \qquad [2.17]$$

That is, $[T^*]$ maps $\overline{N}$ in such a way that the transformed normal vector $[T^*]\overline{N}$ is still orthogonal to the transformed surface. Now since $([T^*]\overline{N})^T = \overline{N}^T[T^*]^T$, [2.17] may be rewritten as

$$\overline{N}^T[T^*]^T[T]\,(\overline{x} - \overline{x}_o) = 0 \qquad \forall\,\overline{x}\,S,\ \overline{x} \neq \overline{x}_o \qquad [2.18]$$

Now [2.18] is obviously true whenever

$$[T^*]^T[T] = k[I] \qquad [2.19]$$

where k is any non-zero scalar. Now, if [T] is of the form

$$[T] = \left[\begin{array}{c|c} [A]_{3x3} & [B]_{3x1} \\ \hline [0]_{3x1} & [I]_{1x1} \end{array}\right]$$

it has an inverse only if $[A]_{3 \times 3}$ is nonsingular. Brief examination of [2.6] - [2.10] reveals that all simple matrix transformations are of this form, providing that the scaling transformation $S(k_x,k_y,k_z)$ has all components > 0. Furthermore, composite transformations are of the type

$$[T] = \prod_{i=1}^{n} [T_i] = [T_1]\,[T_2]\,\ldots\,[T_{n-1}]\,[T_n]$$

$$= \left[\begin{array}{c|c} \sum_{i=1}^{n} [A_i]_{3x3} & \prod_{i=1}^{n} \left(\prod_{j=1}^{i-1} [A_j]_{3x3}\right) [B_i]_{3x1} \\ \hline [0]_{1x3} & [I]_{1x1} \end{array}\right]$$

where $[T_i]$ is a simple matrix transformation and so $[T^{-1}]$ is guaranteed to exist. Thus, [2.19] is equivalent to

$$[T^*]^T = k[T^{-1}] = \frac{k}{|T|} \mathrm{Adj}[T]$$

or, equivalently

$$[T^*] = \frac{k}{|T|} \mathrm{Adj}^T[T] \qquad [2.20]$$

which proves [2.16].

It should be noted that the brightness functions [2.2] and [2.5] involve normalized gradient vectors $\overline{N}/||\overline{N}||$. Unfortunately, [T*] as given by [2.20] is not, in general, a unitary transformation (due to possible unequal scalings k_x,k_y, and k_z) and so the set of gradient vectors $\left\{\overline{N}_i(\overline{x}) = \partial f_i(\overline{x})/\partial x\right\}$ must be normalized after application of [2.15].

Edge data containing only point labels, surface labels, and gradient vector labels are invariant under any composite transformation.

Projection of R^3 into R^2: Consider Figure 24. The point $\overline{P} = (x,y,z)^T$ is in R^3. The viewing window is located in the constant $z = Z_{vw}$ plane. The eye is located at the origin $\overline{0}$. The projection point $\overline{P}' = (x', y', Z_{VW})^T$ of the point $\overline{P}$ onto the viewing window is desired.

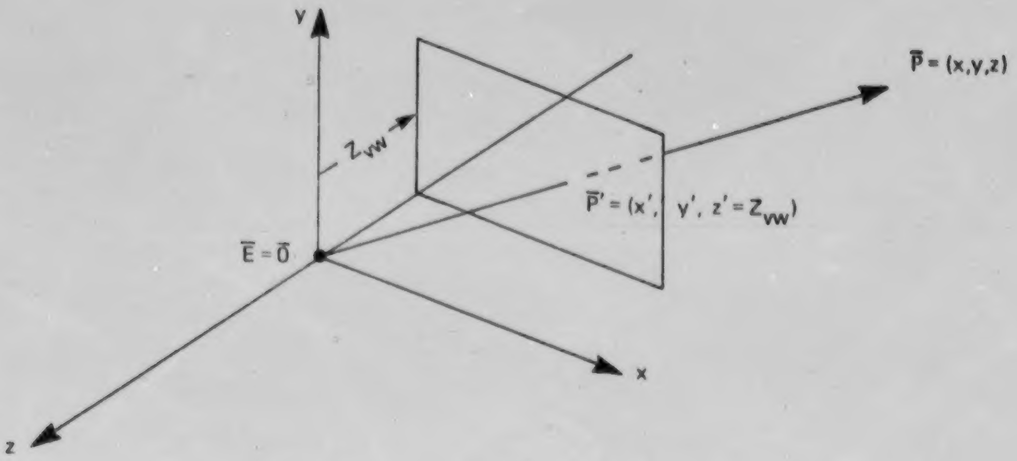

y
Z_{VW}
$\overline{P} = (x,y,z)$
$\overline{P}' = (x', y', z' = Z_{VW})$
$\overline{E} = \overline{0}$
x
z

Figure 24 – **Projection of Point $\overline{P}$ Onto the Viewing Window**

Now, since $\overline{P}' = \alpha\overline{P}$, where α is scalar,

$$\begin{aligned} x' &= \alpha x \\ y' &= \alpha y \\ z' = Z_{VW} &= \alpha z \end{aligned}$$

and from the last, $\alpha = Z_{VW}/z$. That is, α is the ratio of the viewing window depth to the total depth of the point. With α so defined, the first two equations reduce to

$$x' = \frac{Z_{VW}}{z} x \qquad [2.21]$$

$$y' = \frac{Z_{VW}}{z} y \qquad [2.22]$$

Both [2.21] and [2.22] have singularities at $z = 0$. That is, points on a plane that are parallel to the viewing window and that also contain the eye have no valid projection. Also, points with $z < 0$ have no valid projection since they lie "behind" the eye.

Note too from [2.21] and [2.22] that the ratios x'/x and y'/y are greater than unity if $0 < z < Z_{VW}$. That is, an object will be magnified if it lies between the eye and the viewing window. Because of this magnification and the singularities of [2.21] and [2.22], a restriction will be made that objects lie on the "far" side of the viewing window; that is, objects will not be allowed to penetrate the viewing window plane and still be seen. If a surface does penetrate the viewing window, only that part of it which lies on the far side of the viewing window will be displayed.

Now consider the projection of a straight-line edge from R^3 into R^2 as in Figure 25. Since $\overline{P}_1, \overline{P}_2$, and $\overline{E}$ define a plane containing $\overline{P}_1 - \overline{P}_2$ which intersects the plane of the viewing window ($z = Z_{vw}$) in a straight line, [2.21] and [2.22] map straight lines into straight lines.

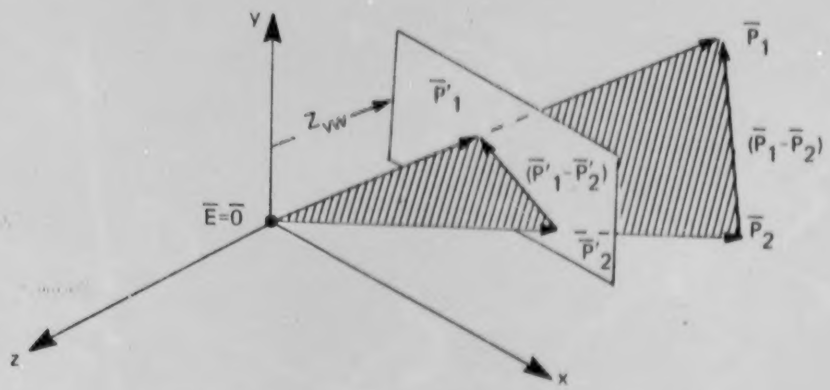

Y
Z_{VW}
$\overline{P}'_1$
$\overline{P}_1$
$(\overline{P}_1 - \overline{P}_2)$
$\overline{E} = \overline{0}$
$(\overline{P}'_1 - \overline{P}'_2)$
$\overline{P}'_2$
$\overline{P}_2$
Z
X

Figure 25 — **Projection of a Straight-Line Edge**

The slope $m = \dfrac{dx}{dy}$ of a projected straight-line edge may be found as a function of the original R^3 point data. m is given by

$$m = \frac{x'_2 - x'_1}{y'_2 - y'_1}$$

where (x'_1, y'_1, Z_{vw}) and (x'_2, y'_2, Z_{vw}) correspond to projected points $\overline{P}'_1$ and $\overline{P}'_2$. Application of [2.21] and [2.22] yields

$$m = \frac{\left(\dfrac{Z_{vw}}{z_2}\right) x_2 - \left(\dfrac{Z_{vw}}{z_1}\right) x_1}{\left(\dfrac{Z_{vw}}{z_2}\right) y_2 - \left(\dfrac{Z_{vw}}{z_1}\right) y_1}$$

$$m = \frac{z_1 x_2 - z_2 x_1}{z_1 y_2 - z_1 y_1} \qquad [2.23]$$

Note that $m = \Delta dx'/dy'$ is the inverse of its normal definition and that here

$$\Delta x' = m \Delta y' \qquad [2.24]$$

and, hence, that the locus of points on a straight-line edge that has been projected onto the viewing window has the form

$$x'(y'_2) = x'(y'_1) + m(y'_2 - y'_1) \qquad [2.25]$$

where $\left(x'(y'_1), y'_1\right)$ and $\left(x'(y'_2), y'_2\right) \in R^2$. Note also that $m \to 0/0$ if both $z_1, z_2 \to 0$. That is, there is no valid projection of edges that lie in the eye plane parallel to the viewing window plane (i.e., the $z = 0$ plane).

Now consider the projection of a planar surface from R^3 into R^2. Since any surface on an atom in R^3 is a planar surface bounded by straight-line edges, the area in R^2 corresponding to that surface is that area bounded by the projected edges. See Figure 26.

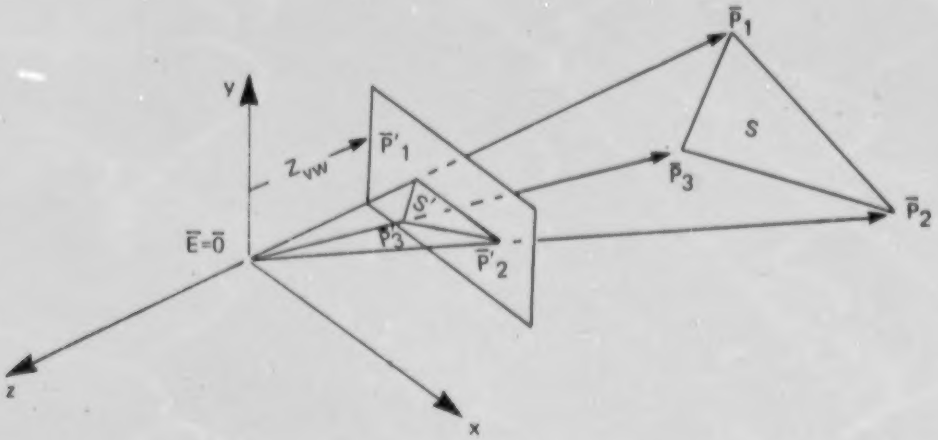


Figure 26 – **Projected Atom Planar Surface S**

Now consider whether or not surface S will be visible after it has been projected into R^2. Three conditions are obvious:

(1) The surface must face toward the eye and not away from it. Let $\overline{N}$ be the gradient vector of planar surface S. Then this condition may be tested at any point $\overline{P}$ on S by

$$\overline{N}^T \cdot (\overline{P} - \overline{E}) = \overline{N}^T \cdot \overline{P} < 0 \qquad [2.26]$$

(2) Some part of the projected surface S′ must lie within the region $\left\{(x', y') \mid x'_{min} \le x' \le x'_{max}, y'_{min} \le y' \le y'_{max}\right\}$ where $x'_{min}, x'_{max}, y'_{min}$ and y'_{max} define the rectangular boundaries of the viewing window in the $z = Z_{vw}$ plane and may be found from [2.14].

(3) Some part of the surface in R^3 must lie on the "far" side of the viewing window. Thus restriction was placed after developing the projection algorithms [2.21] and [2.22]. If all the edges bounding surface S in R^3 lie on the "near" (i.e., $z > Z_{VW}$) side of the viewing window, then S′ will not appear.

A surface S that meets these three conditions is *potentially visible* over the region $\left\{ (x',y') \mid (x',y') \in S' \right\}$ on the viewing window. Whether or not it will be visible at any position $(x',y') \in S'$ depends upon whether or not S is the "nearest" potentially visible surface encountered by an eye sighting from $\overline{E} = \overline{0}$ through that point $(x', y', Z_{VW})^T$ on the viewing window. This distance between a surface and the viewing eye will now be related to the projection of the edges that define the "right" boundaries of S′.

First, consider the relationship in R^2 between projected edges and surfaces. The original atom edge definition $(P_1,P_2,P_3,P_4,S_L,S_R,B_1,B_2,B_3,B_4)$ is such that if one were to travel from end point $\overline{P}_1$ to end point $\overline{P}_2$, he would encounter surface S_L on the left side of the edge and surface S_R on the right side of the edge. If the edge is projected by [2.21] and [2.22] into R^2, there are four possible outcomes:

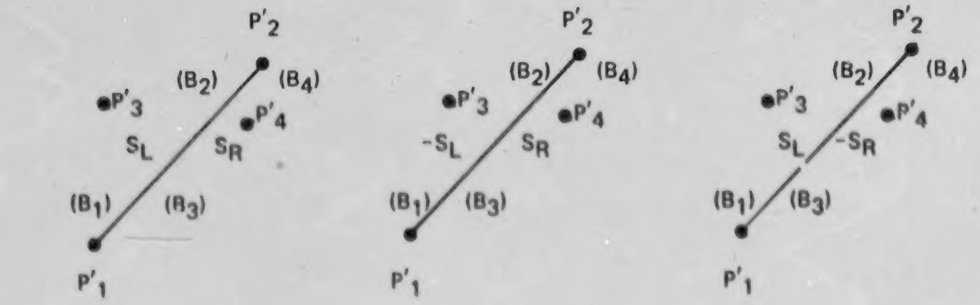
(1) Both S_L and S_R face away from the eye; that is, both surfaces fail to meet Condition (1) for a surface to be potentially visible. This case is that of an edge on the far side of an object. The edge is ignored.

(2) The edge projection in R^2 is horizontal. In this case, one projected surface is above the projected edge and the other projected surface is below the projected edge. Since neither projected surface has any left or right orientation to the edge, the edge is ignored.

(3) At least one of the surfaces meets condition (1) and P'_2, the projection of point $\overline{P}_2$, lies "above" P'_1, the projection of point $\overline{P}_1$. In this case, the original left-right convention remains valid. If either of the surfaces does fail to meet Condition (1), its surface label S is replaced by -S which will flag to the visible surface processor that only the other surface is potentially visible. Figure 27 illustrates this case.

(4) At least one of the projected surfaces meets Condition (1) and $\overline{P}'_2$ the projection of end point P_2, lies "below" $\overline{P}'_1$. In this case, the left-right convention is no longer valid. This may be corrected by interchanging all of the labels as shown in Figure 28. This case then becomes the same as the third outcome.

The relationship between the depth z of a surface in R^3 and one of its "right" boundary edges will now be developed. Consider Figure 29. $\overline{P}_2 = (x_2,y_2,z_2)^T$ has been



S_L and S_R visible $\quad$ S_R visible $\quad$ S_L visible

Figure 27 — **Projected Edge With P'_2 Above P'_1**

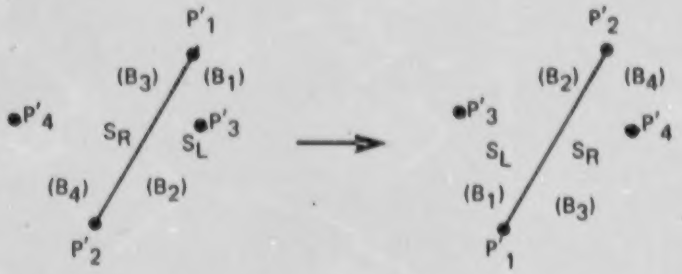


Figure 28 — **Projected Edge With P'_1 Above P'_2**

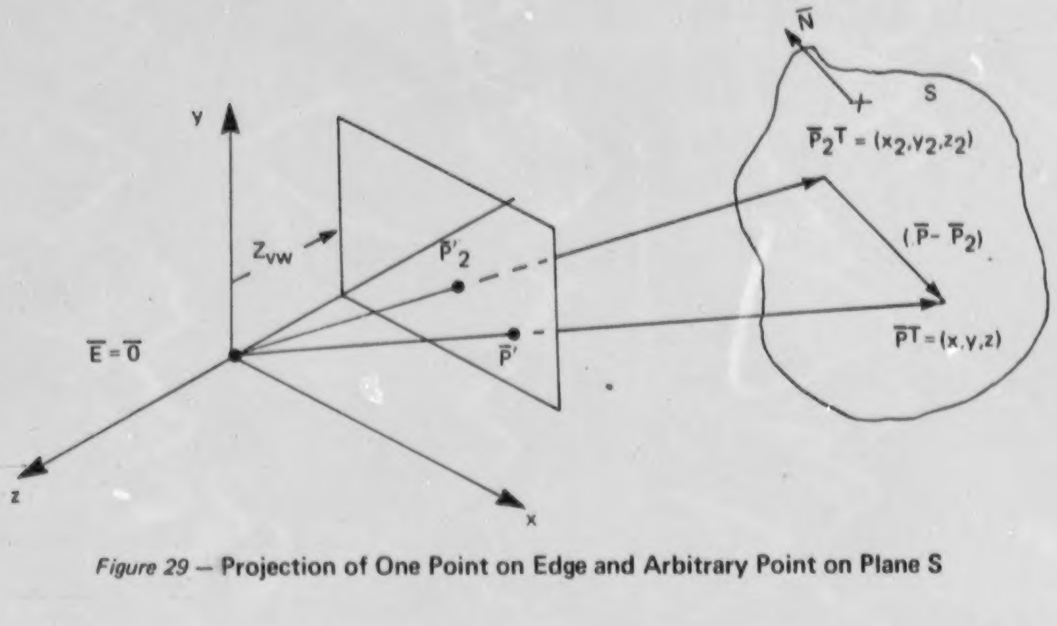


Figure 29 — **Projection of One Point on Edge and Arbitrary Point on Plane S**

determined to produce a projected point $\overline{P}'_2$ which lies "above" P'_1. $\overline{P} = (x,y,z)^T$ is any point on planar surface S. The eye, $\overline{E} = (x_e,y_e,z_e)^T$, is located at the origin $\overline{0}$. The viewing window lies in the $z = Z_{vw}$ plane and $\overline{P}' = (x', y', Z_{vw})^T$ is the projection of $\overline{P}$ onto the viewing window. $\overline{N} = (n_x,n_y,n_z)^T$ is the gradient vector for surface S.

Now, since both $\overline{P}_2$, and $\overline{P}$ are on surface S, the vector $\overline{P} - \overline{P}_2$ lies on S and, hence, is orthogonal to gradient vector $\overline{N}$. Thus,

$$(\overline{P} - \overline{P}_2)^T \cdot \overline{N} = 0 \qquad [2.27]$$

which may be expressed as the scalar equation

$$n_x(x - x_2) + n_y(y - y_2) + n_z(z - z_2) = 0 \qquad [2.28]$$

Also, [2.21] and [2.2] may be solved for the x and y coordinates of the original point $\overline{P}$ in terms of the z component of $\overline{P}$ and the projection point $\overline{P}'$.

$$x = (z/Z_{vw})\,x'$$

$$y = (z/Z_{vw})\,y'$$

substituting into [2.28] and multiplying Z_{vw} yields

$$n_x(z \cdot x' - Z_{vw} \cdot x_2) + n_y(z \cdot y' - Z_{vw} \cdot y_2) + n_z(Z_{vw} \cdot z - Z_{vw} \cdot z_2) = 0$$

$$z(\overline{N}^T \cdot \overline{P}') - Z_{vw}(\overline{N}^T \cdot \overline{P}_2) = 0$$

or simply

$$d = \frac{Z_{vw}}{z} = \frac{\overline{N}^T \cdot \overline{P}'}{\overline{N}^T \cdot \overline{P}_2} \qquad [2.29]$$

For surfaces lying on the "far" side of the viewing window, z ranges from Z_{vw} to $-\infty$ and maps d from +1 to 0^+. Thus, $z \epsilon [Z_{vw}, -\infty)$ implies $d \epsilon [1,0)$. Thus, the problem of determining the nearest surface seen by an eye that is located at the origin and sights through the point $\overline{P}' = (x', y', z' = Z_{vw})$ is that of determining the surface with the maximum d. [2.29] can be written as

$$d = \frac{\overline{N}^T \cdot \overline{P}'}{\overline{N}^T \cdot \overline{P}_2} = \frac{n_x x' + n_y y' + n_z Z_{vw}}{K}$$

$$d = \frac{n_z Z_{vw}}{K} + \frac{n_x}{K} x' + \frac{n_y}{K} y' \qquad [2.30]$$

where [2.30] is a function of the R^2 position (x', y') on the viewing window.

$$d_o = \frac{n_z Z_{vw}}{K} + \frac{n_x}{K} x'_2 + \frac{n_y}{K} y'_2 \qquad [2.31]$$

Thus, if $x' = x'_2 + \Delta x'$ and $y' = y'_2 + \Delta y'$, then

$$\begin{aligned} d &= \frac{n_z Z_{vw}}{K} + \frac{n_x}{K}(x'_2 + \Delta x') + \frac{n_y}{K}(y'_2 + \Delta y') \\ &= d_o + \frac{n_x}{K}\Delta x' + \frac{n_y}{K}\Delta y' \\ d &= d_o + d_x \Delta x' + d_y \Delta y' \qquad [2.32] \end{aligned}$$

where $d_x = n_x/K$ and $d_y = n_y/K$. Thus, [2.32] gives the depth function d for a displacement of $(\Delta x', \Delta y')$ in R^2. Consider now a displacement in R^2 along the projected edge. If the slope of the edge is m, a movement of $\Delta y'$ vertically yields a horizontal movement $\Delta x' = m\Delta y'$ and the depth function is given by

$$\begin{aligned} d &= d_o + d_x(m\Delta y') + dy\ (\Delta y') \\ &= d_o + (d_x m + d_y)\Delta y' \\ d &= d_o + d_s \Delta y' \qquad [2.33] \end{aligned}$$

where $d_s = md_x + d_y$. Thus an expression for the depth function of a projected planar surface where the displacement in R^2 is the composite of $\Delta y'$ vertically along the projected edge, and then $\Delta x'$ horizontally from the projected edge is given by [2.34] and is illustrated by Figure 30.

$$d = d_o + d_x x' + d_s \Delta y' \qquad [2.34]$$

$$d = \frac{\overline{N}^T . \overline{P}_{2'}}{K} \quad (\frac{n_x}{K})\ \Delta x' + (\frac{mn_x + n_y}{K})\ \Delta y' \qquad [2.35]$$

where:

$\overline{N}^T$ = (n_x, n_y, n_z) = Gradient vector to surface S
$\overline{P}_2{}^T$ = (x_2, y_2, z_2) = "Top" end point of edge
$\overline{P}'_2{}^T$ = (x'_2, y'_2, Z_{vw}) = Projected top and point of edge
K = $\overline{N}^T \cdot \overline{P}_2 = n_x x_2 + n_y y_2 + n_z z_2$
m = Slope of projected edge

$\Delta y' = (y' - y'_2)$ = Vertical displacement in R^2 from $\overline{P}'_2$

$\Delta x' = \left(x' - (x'_2 + m\Delta y')\right)$ = Horizontal displacement in R^2 from edge.

Note that in Figure 30 both $\Delta y'$ and $\Delta x'$ are negative.

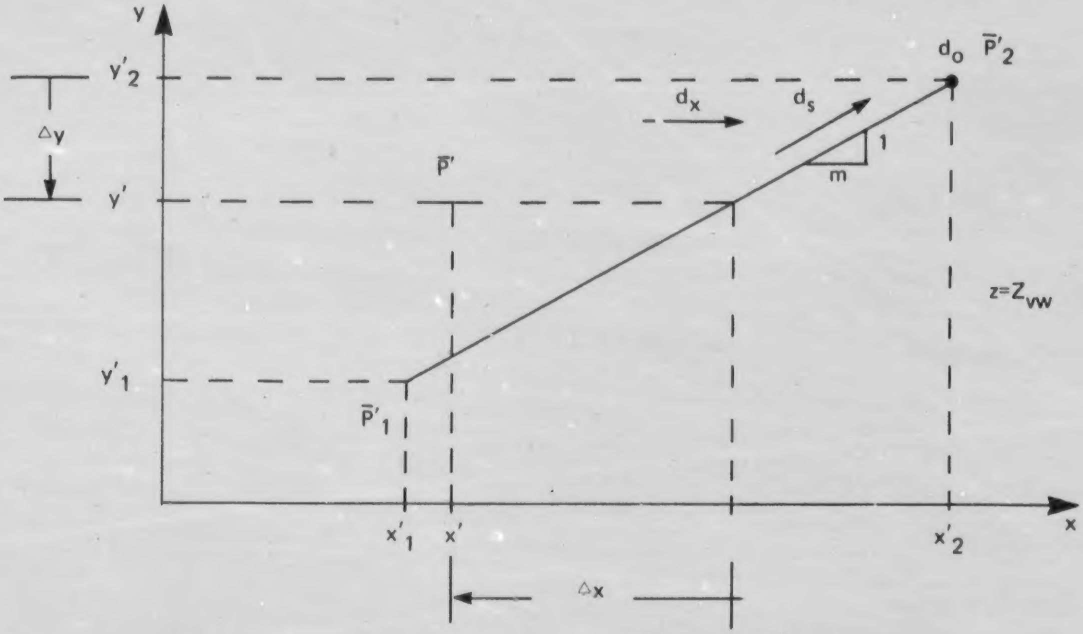


Figure 30 – **Depth at Arbitrary Point $\overline{P}'$**

The final function defined over R^3 which is to be projected onto the R^2 viewing window is the brightness function $B(\overline{x})$ given by [2.36] (derived from [1.11] and [2.14a].)

[2.36]

$$B(x) = BB_{omax}B_F \left(B_S f\left(\frac{\overline{S}^T \cdot \overline{N}(\overline{x})}{\|N(x)\|} \right) + \sum_{i=1}^{P} \frac{B_i}{\left(\frac{\|\overline{x}_i - \overline{x}\|}{r_{o_i}} \right)^2 + 1} \; f\left(\frac{(\overline{x}_i - \overline{x})^T \cdot \overline{N}(\overline{x})}{\|\overline{x}_i - \overline{x}\| \; \|\overline{N}(\overline{x})\|} \right) \right)$$

where:

B_o: Coefficient of optical transmission for observer; $B_o \in [0,1]$.

B_{max}: Maximum allowed brightness of any one source; $B_{max} > 0$.

B_F: Coefficient of optical reflectivity of surface; $B_F \in [0,1]$.

$f(\alpha)$: Reflectivity function.

B_S: Fraction of B_{max} that serves as brightness of sun light source; $B_S \in [0,1]$.

$\overline{S}$: Direction of the sun's light rays.

$\overline{N}(\overline{x})$: Gradient vector to surface at point $\overline{x}$.

P: Number of point light sources.

B_i: Function of B_{max} that is the i^{th} light source's maximum brightness; $B_i \in [0,1]$.

$\overline{x}_i$: Location in R of the i^{th} light source.

r_{o_i}: A constant such that $\| \overline{x}_i - \overline{x} \| / r_{o_i}$ is a brightness fall-off ratio for the i^{th} light source; $r_{o_i} > 0$.

There are several possible brightness definitions that an edge may have:

(1) Both the left and right surfaces are "flat." For this case, $\overline{N}_L$ may serve for both points $\overline{P}_1$ and $\overline{P}_2$ on the left surface and $\overline{N}_R$ may serve for both $\overline{P}_1$ and $\overline{P}_2$ for the right surface.

SPECIAL CASE: If (1) holds *and* if $P = 0$, then the brightness of each surface is a constant independent of $\overline{x}$ (see [1.13]. These surfaces will be called *flat-shaded surfaces*.

(2) The edge is to be smoothly shaded across (that is, $B(\overline{x})$ is continuous across the edge) and $\overline{N} = (\overline{N}_L + \overline{N}_R) / \| \overline{N}_2 + \overline{N}_R \|$ defines the gradient vector at any point $\overline{x}$ on the edge. In this case there is one brightness at $\overline{P}_1$ and a second brightness at $\overline{P}_2$.

SPECIAL CASE: If (2) holds *and* if $P = 0$, then the brightness along the edge is constant.

(3) The edge is to be smoothly shaded across with $\overline{N}_1$ defining the brightness normal at $\overline{P}_1$ and $\overline{N}_2$ defining the brightness normal at $\overline{P}_2$. This is the typical form of an edge on a planar surface facet that is used to approximate a curved surface over some region.

(4) The edge is either flat-to-curved, curved-to-flat, or curved-to-curved. These edges are typically the result of the intersection of two atom surface constraints, for example, the junction of the flat ends and the curved sides of a right circular cylinder. The brightness function is discontinuous over these edges. Each of these edges generates two projected edges: one for the left-side surface using B1 and B2, and one for the right-side surface using B3 and B4.

Thus, the brightness of an atom edge is typically specified at its end points by application of [2.36] to the brightness normals at the appropriate end points. Since Case (4) generates two projected edges, the general case may be taken to be that of Figure 31 without loss of generality.

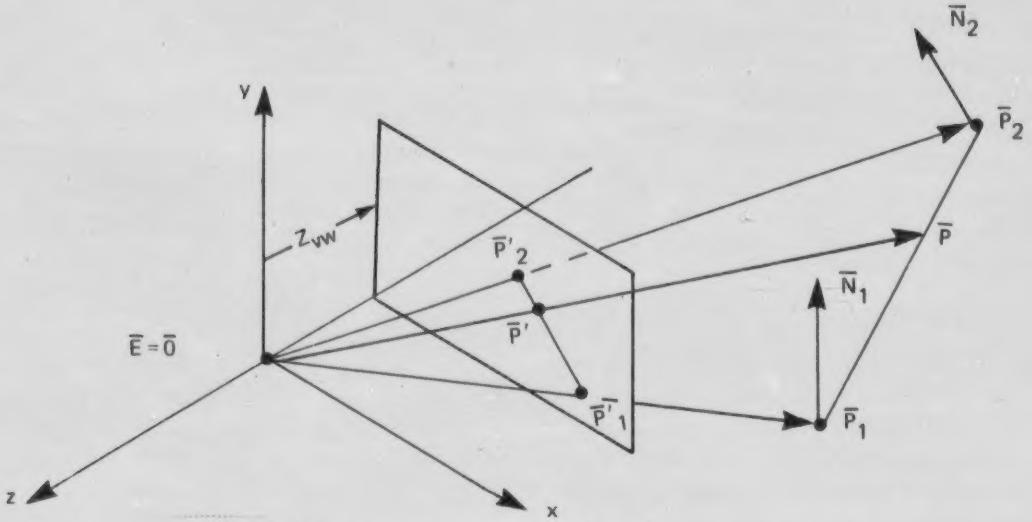


Figure 31 – **Projection of an Edge With Two Brightness Normals**

The continuous brightness function along the edge had been defined as the logarithmic function (see [1.15]).

$$B(\lambda) = B(\overline{P}_2)\left(\frac{B(\overline{P}_1)}{B(\overline{P}_2)}\right)^{\lambda}$$

where $\lambda \in [0,1]$ represents the fractional distance of $\overline{x}$ from $\overline{P}_2$ to $\overline{P}_1$; that is, $\overline{P} = \lambda(\overline{P}_1) + (1-\lambda)\overline{P}_2$. However, $\overline{P}' \neq \lambda\overline{P}'_1 + (1-\lambda)\overline{P}'_2$ and, so, all smooth shading will be done in the perspective domain approximating the shading in R^3.
Thus, since in R^2 an edge has the form

$$x'(y') = x'(y'_2) + m\,(y' - y'_2)$$

which is linear in y′, the approximating brightness can be written as

$$B(y') = B(y'_2) \cdot \left(\frac{B(y'_1)}{B(y'_2)}\right)^{\frac{y' - y'_2}{y'_1 - y'_2}} \qquad [2.37]$$

where

$$B(y'_2) = B(\overline{P}'_2) = B(\overline{P}_2)$$

$$B(y'_1) = B(\overline{P}'_1) = B(\overline{P}_1)$$

For computational purposes, the form of [2-37] may be altered. Taking the logarithm of both sides of [2.37]

$$\log_a \left(B(y')\right) = \log_a \left(B(y'_2)\right) + \frac{(y' - y'_2)}{(y'_1 - y'_2)} \left[\log_a \left(B(y'_1)\right) - \log_a \left(B(y'_2)\right)\right]$$

or, more simply

$$b = b_o + (y' - y'_2)g \qquad [2.38]$$

where

$$b_o = \log_a \left(B(y'_2)\right) = \ell n\, B(y'_2) \,/\, \ell n\; a \qquad [2.39]$$

$$g = \frac{\left[\log_a \left(B(y'_1) - \log_a \left(B(y'_2)\right)\right]\right.}{(y'_1 - y'_2)} = \frac{\left[\ell n\, B(y'_1) - \ell n\, B(y'_2)\right]}{(y'_1 - y'_2)\ell n\; a} \; . \qquad [2.40]$$

Thus, the brightness function along an edge may be put into a form identical to that of [2.33], the depth function d along the edge.

The video output at any point $\overline{P}' = (x', y', z_{vw})^T$ on projected surface S is given by the vector

$$\overline{V}\,(\overline{P}') = B\,(\overline{P}')\,\overline{C}'\,(S) \qquad [2.41]$$

where $\overline{C}'(S) = (c'_g, c'_b, c'_r)^T$ defines the transformed color of surface S. Then

$$\log_a\left(\overline{V}(\overline{P}')\right) = \log_a \left(B(\overline{P}')\,\overline{C}'(S)\right) \qquad [2.42]$$

or, more simply

$$\overline{v}(\overline{P}') = \overline{b}(\overline{P}') + \overline{c}(S) \qquad [2.43]$$

where $\overline{v}$ and $\overline{c}$ are logarithmic vector functions and $\overline{b}(\overline{P}') = \left(b(\overline{P}'), b(\overline{P}'), b(\overline{P}')\right)^T$. Equation [2.43] has the scalar form

$$\begin{aligned} \text{green } (\overline{P}') &= b(\overline{P}') + c_g \\ \text{blue } (\overline{P}') &= b(\overline{P}') + c_b \\ \text{red } (\overline{P}') &= b(\overline{P}') + c_r \end{aligned} \qquad [2.44]$$

where c_g, c_b, and c_r are logarithmic measures of c'_g, c'_b, and c'_r. Of course, the terminal must perform the antilog functions

$$\overline{V}(\overline{P}') = \log_a^{-1}\left(\overline{v}(\overline{P}')\right) \qquad [2.45]$$

or, for each of the three color components,

$$V_i(\overline{P}') = a^{b(\overline{P}')+c_i} \qquad i = 1,2,3 \qquad [2.46]$$

This is most easily done by giving the three video amplifiers an exponential (i.e., anti-logarithmic) response.

The value of logarithmic base "a" is calculated in Appendix B.

Window Clipping: X and Y

The three requirements for a plane surface S in R^3 to be potentially visible in R^2 are:

(1) The surface must face toward the eye.

(2) Some part of the projected surface must lie in the region $\{(x',y') \mid x'_{min} \leq x' \leq x'_{max}, y'_{min} \leq y' \leq y'_{max}\}$ where x'_{min}, x'_{max}, y'_{min} and y'_{max} define the rectangular viewing window in the $z = Z_{vw}$ plane.

(3) Some part of the surface S in R^3 must lie on the far side of the viewing window.

Condition 1 is satisfied when edge parameters are projected onto the plane of the viewing window. That procedure is described in Section 1, Atom Projection Loop (pg. 45). This section describes implementation of Conditions 2 and 3.

The first step involved in the clipping process is to reference the projected edge parameters to a more convenient set of axes. Consider Figure 32. The original viewing window is shown relative to the original x' and y' axes and $z = Z_{vw}$. A new set of axes X and Y is drawn such that X increases positively to the left of x'_{max}, and Y' increases positively downward from y'_{max}.

Also corresponding to the viewing window is a resolution grid structure of R_x elements in x by R_y elements in y. For standard television R_x=640 and R_y=480 yielding the standard aspect ratio R_x/R_y=4/3. For correct (non-distorted) imaging, the observer's viewing window should be specified such that W_x/W_y=4/3. For the CHARGE terminal R_x=1600 and R_y= 1200 yielding R_x/R_y=4/3 as its aspect ratio if W_x/W_y=4/3. For non-standard video display formats, the grid resolution should reflect the W_x/W_y ratio. For example, suppose only the top-half of the television screen is to be used for a particular image. Then W'_x/W'_y=4/1.5. For the resulting image to be unwarped, R'_x/R'_y=4/1.5 or R'_x=1600 (for full horizontal image on the CHARGE terminal) and R'_y=600. The resulting display would have projected objects of the same size and shape as the original display, but the top and bottom quarters of the original display would no longer be visible.

Figure 33 demonstrates one possible implementation of W_x/W_y=4/1.5. The unused scanlines (1/2(1200)=600) were defined to be below the used scanlines; these unused scanlines may then be used to display a second image concurrently with the first.

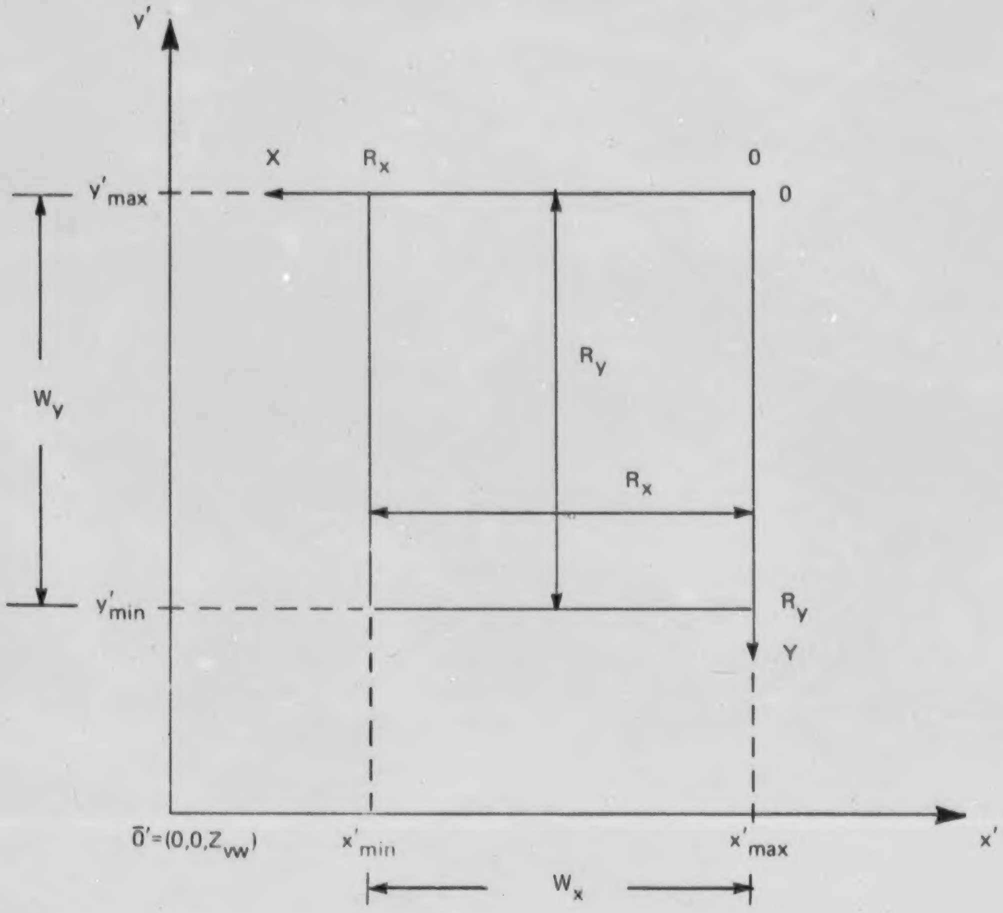
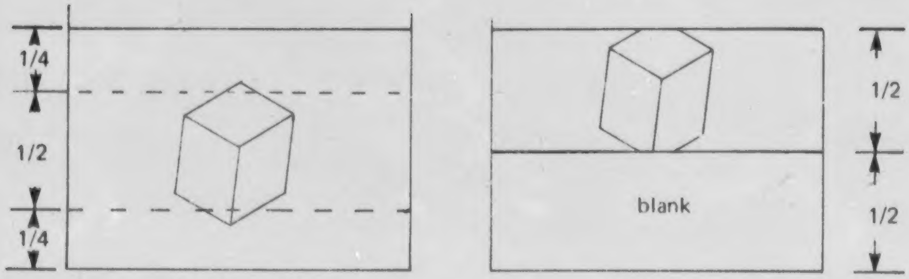


Figure 32 – **Translated Axes X and Y With Corresponding Viewing Window**



$W_x/W_y = 4/3$ $W_x/W_y = 4/1.5$

Figure 33 – **Effect of $W_x/W_y \neq 4/3$ ($R_x/R_y = W_x/W_y$**

The convention that will be used throughout the remainder of this section is that the viewing window is W_x units (meters) wide by W_y units high and that the image is to be displayed over a grid structure of R_x resolution points horizontally by R_y scanlines vertically. For the CHARGE terminal, $R_x \leqslant 1600$ and $R_y \leqslant 1200$. With this convention, the line $x' = x'_{max}$ maps into the line $X = 0$; the line $x' = x'_{min}$ maps into the line $X = R_x$; the line $y' = y'_{max}$ maps into the line $Y = 0$; and the line $y' = y'_{min}$ maps into the line $Y = R_Y$.

The general rule for mapping points is

$$X = \frac{R_x}{W_x} (x'_{max} - x') = X_o + ax' \qquad [2.47]$$

$$Y = \frac{R_y}{W_y} (y'_{max} - y') = Y_o + by' \qquad [2.48]$$

Slopes of projected edges are now given by

$$M = \frac{\Delta x}{\Delta y} = \frac{(X_o + ax'_1) - (X_o + ax'_2)}{(Y_o + by'_1) - (Y_o + by'_2)}$$

$$= \frac{a(x'_1 - x'_2)}{b(y'_1 - y'_2)} = \frac{a}{b} m$$

$$M = \frac{R_x W_y}{R_y W_x} m = cm \qquad [2.49]$$

If $(R_x/R_y) = (W_x/W_y)$, as is always the case for an undistorted picture (partitioning of the window only), then

$$M \Big|_{(R_x/R_y) = (W_x/W_y)} = m \qquad [2.50]$$

The incremental depth function along a horizontal line is given by

$$D_x = \frac{D(X_1) - D(X_2)}{X_1 - X_2}$$

$$= \frac{d(X'_1) - d(X'_2)}{X_1 - X_2}$$

$$= \frac{d(X'_1) - d(X'_2)}{a(X'_1 - x'_2)}$$

$$D_x = \frac{1}{a} d_x \qquad [2.51]$$

The incremental depth function along a projected edge is

$$D_s = \frac{D(Y_1) - D(Y_2)}{Y_1 - Y_2}$$

$$= \frac{d(y'_1) - d(y'_2)}{Y_1 - Y_2}$$

$$= \frac{d(y'_1) - d(y'_2)}{b(y'_1 - y'_2)}$$

$$D_s = \frac{1}{b} d_s \qquad [2.52]$$

Obviously, $B(X,Y) = B(x', y')$. The gradient of brightness along an edge is now given by

$$G = \frac{[\log_a B(\bar{P}'_1) - \log_a B(\bar{P}'_2)]}{Y_1 - Y_2}$$

$$G = \frac{1}{b} g \qquad [2.53]$$

All of the edge parameters have now been converted to the new axis system.

EXAMPLE: Let $W_x = 1/3$ meter, $W_y = 1/4$ meter and let $R_x = 1600$ and $R_y = 1200$ (full screen display). Then $W_x/W_y = 4/3 = R_x/R_y$ (image is undistorted). Also,

$a = -R_x/W_x = -4800$ resolution pts/meter

$b = -R_y/W_y = -4800$ scanlines/meter

$m = k$ meters/meter $=> M = k$ resolution pts/scanline

d_x units/meter → $D_x = \dfrac{-d_x}{4800}$ units/resolution pt

d_s units/meter → $D_g = \dfrac{-d_s}{4800}$ units/scanline

G units/meter → $g = \dfrac{-G}{4800}$ units/scanline

Constants that are valid for the entire image and which may be evaluated during initial processing are:

$$x'_{max} = \frac{W_x}{2} - MAG \cdot X'_E$$

$$y'_{max} = \frac{W_y}{2} - MAG \cdot Y'_E \qquad [2.54a]$$

$$a = -R_x/W_x \rightarrow a^{-1} = 1/a$$

$$\rightarrow c = a \cdot b^{-1}$$

$$\begin{aligned} b &= -R_y/W_y \rightarrow b^{-1} = 1/b \\ X_o &= a/x'_{max} \\ Y_o &= b \cdot y'_{max} \end{aligned} \qquad [2.54b]$$

Initial processing of atom edge parameters yields:

$$\begin{aligned} K &= \overline{N}^T \cdot \overline{P}_2 = n_x \cdot x_2 + n_y \cdot y_2 + n_z \cdot z_2 \\ \overline{P}'_1 &= (x'_1, y'_1, z'_1)^T = \left(\frac{Z_{vw}}{z_1} \cdot x_1, \frac{Z_{vw}}{z_1} \cdot y_1, Z_{vw}\right)^T \\ \overline{P}'_2 &= (x'_2, y'_2, z'_2)^T = \left(\frac{Z_{vw}}{z_2} \cdot x_2, \frac{Z_{vw}}{z_2} \cdot y_2, Z_{vw}\right)^T \\ m &= \frac{x'_2 - x'_1}{y'_2 - y'_1} = \frac{z_1 \cdot x_2 - z_2 \cdot x_1}{z_1 \cdot y_2 - z_2 \cdot y_1} \end{aligned} \qquad [2.55]$$

$$B(\overline{P}'_j) = B_o B_{max} B_F \left[\left(B_a + B_s\, f\left(\frac{\overline{S}^T \cdot \overline{N}(\overline{x}_j)}{\|\overline{N}(\overline{x}_j)\|}\right) + \sum_{i=1}^{P} \frac{B_i}{\left(\frac{\|\overline{x}_i - \overline{x}_j\|}{r_i}\right)^2 + 1}\, f\left(\frac{(\overline{x}_i - \overline{x}_j)^{\overline{T}} \cdot \overline{N}(\overline{x}_j)}{\|\overline{x}_i - \overline{x}_j\| \quad \|\overline{N}(\overline{x}_j)\|}\right)\right)\right] \quad j=1,2$$

where K also serves as test [2.26] as to whether or not the surface faces toward the eye. Final processing of the edge parameters yields:

$$\begin{aligned} X_i &= ax'_i + X_o && i = 1,2 \\ Y_i &= by'_i + Y_o && i = 1,2 \\ M &= cm \\ D_o &= \frac{\overline{N}^T \cdot \overline{P}_2}{K} = d_o \\ D_X &= a^{-1} \cdot \frac{nx}{K} = a^{-1} \cdot d_x \\ D_s &= b^{-1} \cdot \frac{(mn_x + n_y)}{K} = b^{-1} \cdot d_s \\ b(Y_i) &= \log_a\left(B(\overline{P}'_i)\right) && i = 1,2 \\ g &= b^{-1}[b(Y_1) - b(Y_2)] / (y'_1 - y'_2) \end{aligned} \qquad [2.56]$$

Now consider the transformed projection of an edge onto the $z=Z_w$ plane. The viewing window R is $\{(X,Y) \mid 0 < X \leqslant R_x,\ 0 < Y \leqslant R_y\}$. The goal of the X-Y clipping unit is to restrict the set of transformed projected edges to a subset that might generate projected surfaces $\{S'\}$ such that $S'_i \in R$. There are several possible cases vertically:

(1) If $Y_1 \leqslant 0$, the entire projected edge lies above the viewing window (since $Y_2 < Y_1$). Since an edge serves as a boundary between one surface on its left side and another surface on its right side, the edge may be deleted from the projected atom edge set.

(2) Similarly, if $Y_2 > R_y$, the edge lies entirely below the viewing window and the edge may be deleted from the projected atom edge set.

(3) If an edge extends below $Y = R_y$, it may be arbitrarily defined to end at $Y = R_y$.

(4) If an edge starts above the viewing window but extends into the region $1 \leqslant Y \leqslant R_y$, it may be arbitrarily started at $Y = 1$. This means

$$\begin{aligned} Y'_2 &= 1 \\ X'_2 &= X_2 + M(1-Y_2) \\ D_o &= D_o + D_s(1-Y_2) \\ b_{top} &= b_{top} + g(1-Y_2) \end{aligned}$$

Figures 34 and 35 demonstrate how edges are clipped vertically into the proper Y range.

Edges that survive the vertical clipping operation are clipped into the proper horizontal X range. It should be remembered that edges serve three functions: (a) they form left-right partitions between projected atom surfaces; (b) they define the depth functions and the color of the left-side projected surface; and (c) they define the brightness along the perimeters of projected atom surfaces. Thus, if the set of projected atom edges that cross a Scanline Y_i were to be scanned in order of increasing X, each edge encountered would serve to (a) "end" the surface on its right side, (b) "start" a surface on its left side, and (c) define a known brightness to be used by the logarithmic brightness function [1.15] within the surface perimeters. These are operations that the visible surface processor will have to perform.

The first operation, X-ordering a set of projected edges, is impossible to accomplish via one-pass techniques if the range of possible X values is an infinite set. Furthermore, the only surfaces that will be seen lie in the interval $X \in [0, R_x]$. Accordingly, each

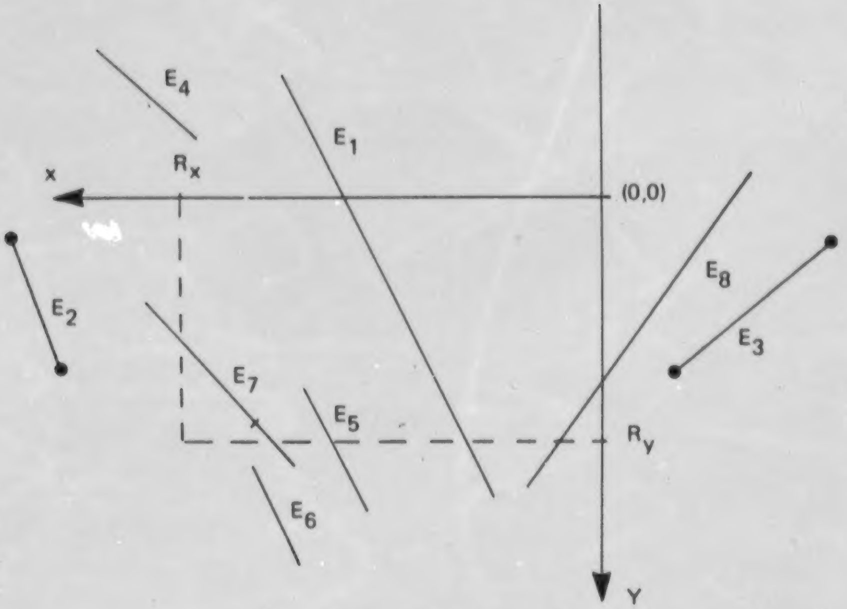


Figure 34 – **Unclipped Projected Edges**

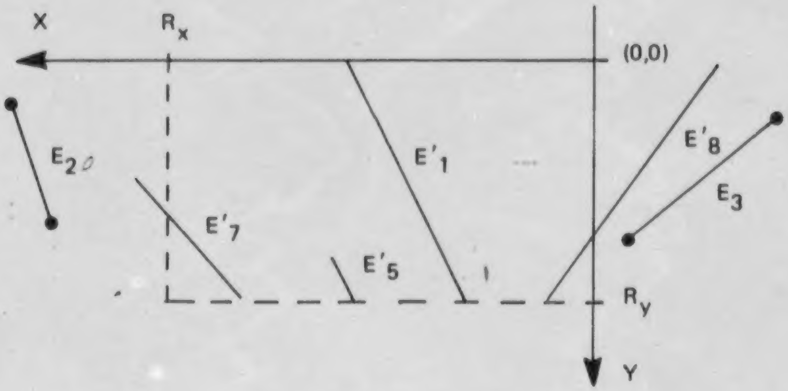


Figure 35 – **Edges Clipped in Y**

projected edge is assigned a fixed point X with indicator value I_X to be set each time the edge crosses a scanline.

$$I_X = \begin{cases} 0 & X_2 \leqslant 0 \\ n & n-1 < x_2 \leqslant n, \quad 0 < n < R_x \\ R_X & X_2 \geqslant R_x \end{cases} \qquad [2.57]$$

Utilization of I_X for X-ordering reduces the infinite set $\{X \mid -\infty < X < +\infty\}$ to the finite set $\{0, 1, 2, \ldots, R_X\}$.

If both $X_1 > R_X$ and $X_2 > R_X$, the entire edge lies to the left of the viewing window and $I_X = R_X$ along the edge. (a) If the right-side surface of the edge is a "flat-shaded" surface (i.e., the surface is planar and there are no point light sources in the world), the edge may be deleted from the set of projected atom edges since it would only be used to turn off a right-side surface of known constant brightness. That is, the left-side surface is not in the pyramid of vision and [1.15] is a constant. (b) If the right-side surface is "smooth-shaded," the edge must be kept to provide a brightness value and an X value for [1.15]. The edge's depth functions (for the surface on the left) are meaningless.

Similarly, if either $X_1 > R_X$ and $0 \leq X_2 \leq R_X$ or if $X_2 > R_X$ and $0 \leq X_1 \leq R_X$, the edge is used only for brightness data over the scanlines for which $I_X = R_X$.

If $X_1 > 0$ and/or $X_2 < 0$, the edge must be modified. A new edge with $X'_1 = X'_2 = 0 = I_X$ is created to represent the surface (which was started to the right of the viewing window by the original edge) along the right side of the viewing window. The depth functions become

$$\begin{aligned} D'_o &= D_o + D_X(0 - X_2) \\ D'_s &= D_Y = D_s - M \cdot D_x \end{aligned} \qquad [2.58]$$

The original X and M are kept for the brightness algorithm [1.15]. Figures 36 and 37 demonstrate the effects of horizontal clipping. Edges E_1 and E_4 lie to the left of the viewing window; the right-side surface of E_1 is "smooth-shaded" while the right-side surface of E_4 is "flat-shaded." Accordingly, E_4 is deleted from the final edge set while E_1 is kept for possible brightness use. Edge E_2 lies to the right of the viewing window. A new edge with $X = 0$ and depth functions [2.58] is created to start the depth functions for the surface at the minimum $X = 0$; the original X and M are still retained for possible brightness use. Edge E_3 extends both to the left and the right of the viewing window and generates two clipped edges. The first generated Edge E'_3 is at first useful for brightness only since $I_X = R_X$ but then is useful for both brightness and surface depth. The second generated Edge E''_3 provides accurate depth functions for its left-side surface at $X = 0$, but retains the original edge slope for brightness use.

Two clipping flags are also provided in the clipping section. Both flags are initially set to +1. The "left" flag, I_L, is cleared to 0 if a projected edge has $X_2 > 0$. The "right"

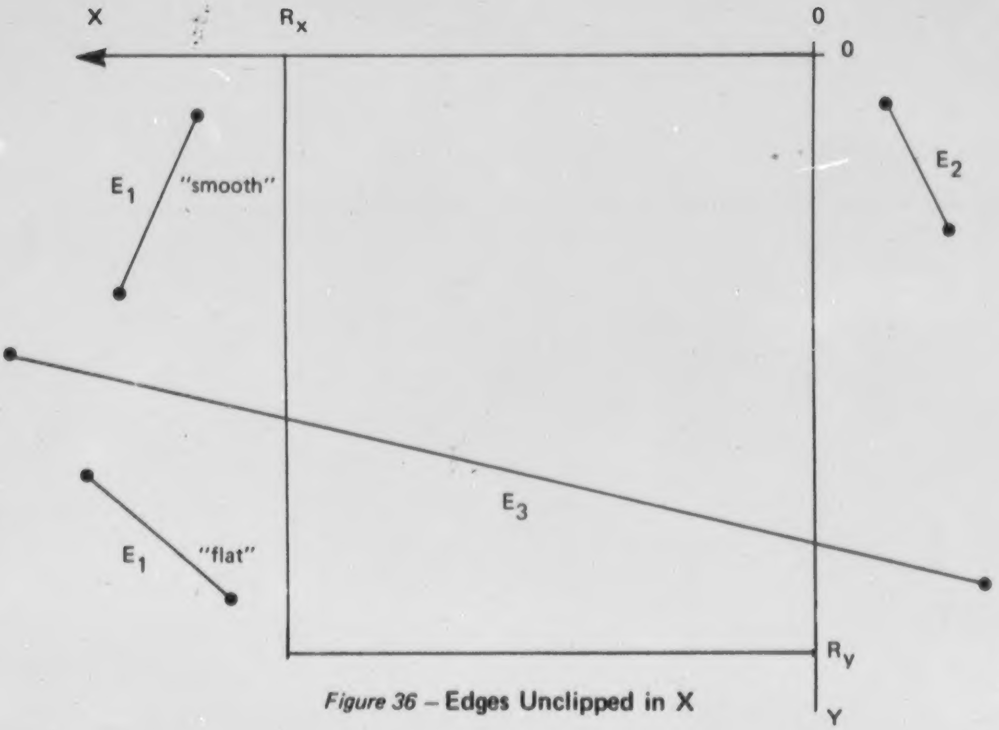


Figure 36 – **Edges Unclipped in X**

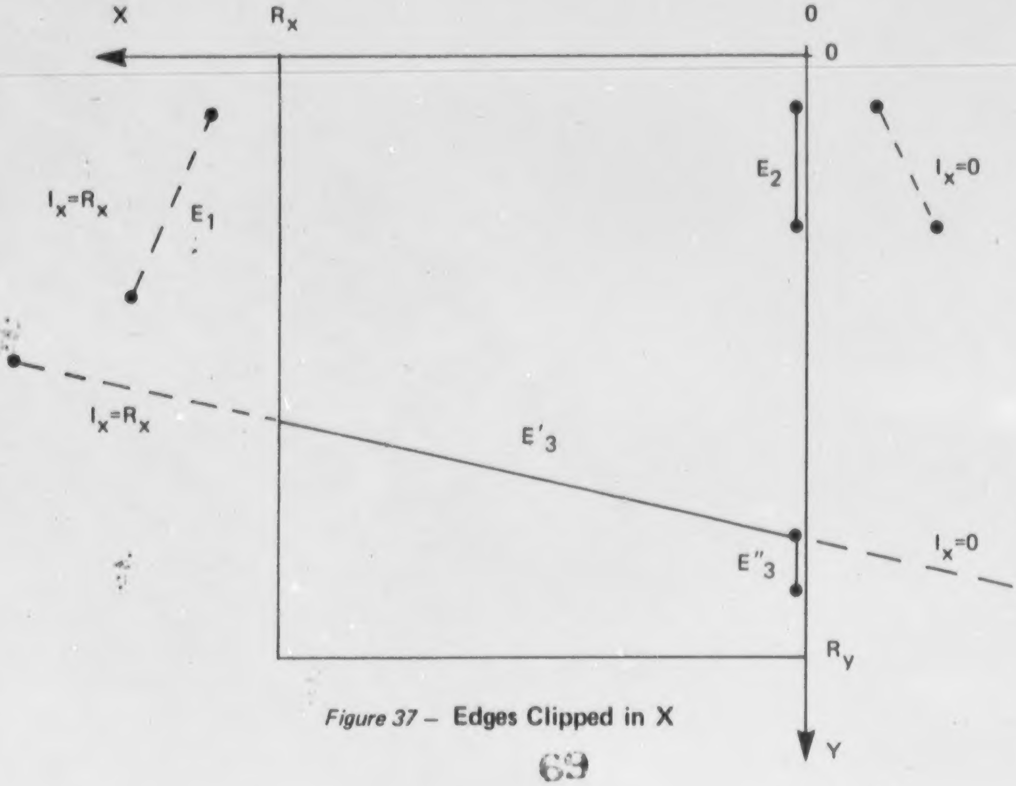


Figure 37 – **Edges Clipped in X**

flag, I_R, is cleared to 0 if a projected edge has $X_2 < R_X$. After all atom edges have been projected, the function $I = I_L + I_R$ is examined. Possible outcomes are:

(1) $I=0$: The projected atom surfaces are potentially visible. Add the projected atom edges to the y-ordered output data set for the visible surface processor.

(2) $I=1$: Either $I_L=1$ or $I_R=1$ and the projected atom lies either entirely to the left of the viewing window or entirely to the right of the viewing window. Delete the entire projected atom edge set.

(3) $I=2$: Both $I_L=1$ and $I_R=1$ indicating that all the atom edges failed to survive the y-clipping. The set of projected atom edges is empty.

Window Clipping: Z

Suppose an object edge in R^3 has one end-point on the near (eye) side of the viewing window and the other end-point on the far side. There are singularities at $z=0$ in the projection operators [2.21] and [2.22], the projected edge slope [2.23], and the depth function [2.29]. Furthermore, if $z \geqslant Z_{VW}$, the range of the depth function is limited to [1,0). Therefore, projected objects shall be clipped to $z \geqslant Z_{VW}$ much as they were clipped to an R_x by R_y viewing area.

The problem of clipping edges at $z=Z_{VW}$ is somewhat similar to that of clipping edges on the right side of the viewing window at $X=0$. The portion of the edge corresponding to $z \geqslant Z_{VW}$ may be projected into R^2 and treated as a normal edge to be clipped in X and Y. The portion of the edge corresponding to $z < Z_{VW}$ may be ignored; however, this portion of the edge may generate a surface on its left side which extends into $z \geqslant Z_{VW}$. In order to start this surface and possibly to turn off a right-side surface, second and third edges are created. These new edges lie in the plane of the viewing window.

Consider Figure 38, the intersection of plane ABCD and the plane of the viewing window, $z=Z_{VW}$. The original plane is defined by the four edges: AB, BC, CD, and DA. It is readily apparent that:

(1) Two different planes will either intersect along a straight-line ℓ or will not intersect at all. Therefore, since all atom edges defining a planar surface lie on that surface, all intersections of those edges and the plane of the viewing window will lie along some straight-line ℓ.

(2) Since all atom edges are terminated at points which are finite in R^3 and, since the edges form a closed boundary for the surface, there will either be no intersections or an even number of intersections between atom edges and the plane of the viewing window for each surface. Since the new edges formed between the points of

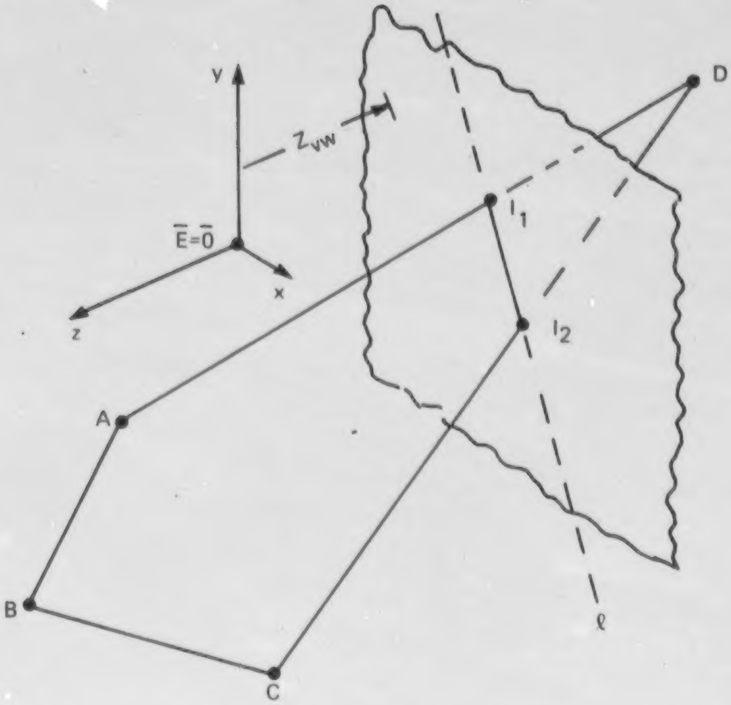


Figure 38 – **Intersection of Plane ABCD and Viewing Window Plane**

intersection between the edges and the viewing window plane complete the boundary of the surface on the far side of the screen, all edges such as AB and BC which lie entirely on the near side of the viewing window plane may be deleted.

(3) If ℓ is horizontal, all of these new edges lying in the viewing window plane are horizontal and may be deleted; that is, projected edges of this type have no left or right surfaces.

(4) If ℓ is not horizontal, there is a unique y value for each point of intersection and the new edges may be formed by starting at the greatest y value and connecting successive y intersections. Figure 39 demonstrates this procedure for a planar surface which has a "W" shape. Edges AB, CD, and EF are normal atom edges. Edge IJ lies entirely on the near side of the viewing window and is ignored. Edges GC, HE, IA, HD, IF, and GB intersect the viewing window plane $z=Z_{vw}$ at I_1, I_2, I_3, I_4, I_5, and I_6. The points of intersection are y-ordered and new Edges I_3I_6, I_1I_4, and I_2I_5 complete the planar surface's boundaries.

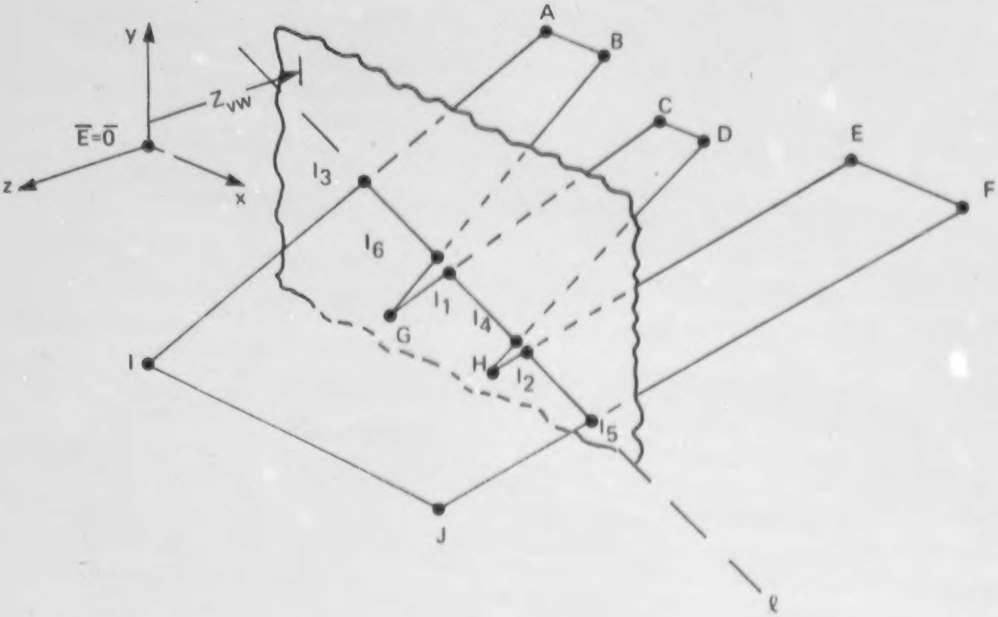


Figure 39 – **Intersection of General Planar Surface With z = Z_W Plane**

The equation of line ℓ can easily be determined from the surface normal, $\overline{N}$, and a point, $\overline{P}_2$, known to be on the surface. Let S denote the planar surface. Let $\overline{P}$ be any point on line ℓ, but let $\overline{P} \neq \overline{P}_2$ as in Figure 40.

Now since $\overline{P}_2 \in S$ and $\overline{P} \in S$, $(\overline{P} - \overline{P}_2)$ is orthogonal to $\overline{N}$ and so

$$(\overline{P}-\overline{P}_2)^T \cdot \overline{N} = 0$$

$$(x-x_2)n_x + (y-y_2)n_y + (Z_{vw}-z_2)n_z = 0$$

which yields

$$x = \left(-\frac{n_y}{n_x}\right)y + \left(x_2 + \frac{n_y}{n_x}\,y_2 + \frac{n_z}{n_x}\,(z_2 - Z_{vw})\right) \qquad [2.59]$$

which has the form of a straight line, x=my+c. Since all points of intersection between S and the plane of the viewing window satisfy [2.59], the slope of ℓ ($m = -n_y/n_x$) and the constant ($c = x_2 + (n_y/n_x)y_2 + (n_z/n_x)(z_2 - Z_{vw})$) need be calculated only once for each planar surface. Note also that m and c are undefined only if $n_x = 0$. This occurs only if S and the plane of the viewing window intersect along a horizontal line. Such horizontal cases are deleted as before.

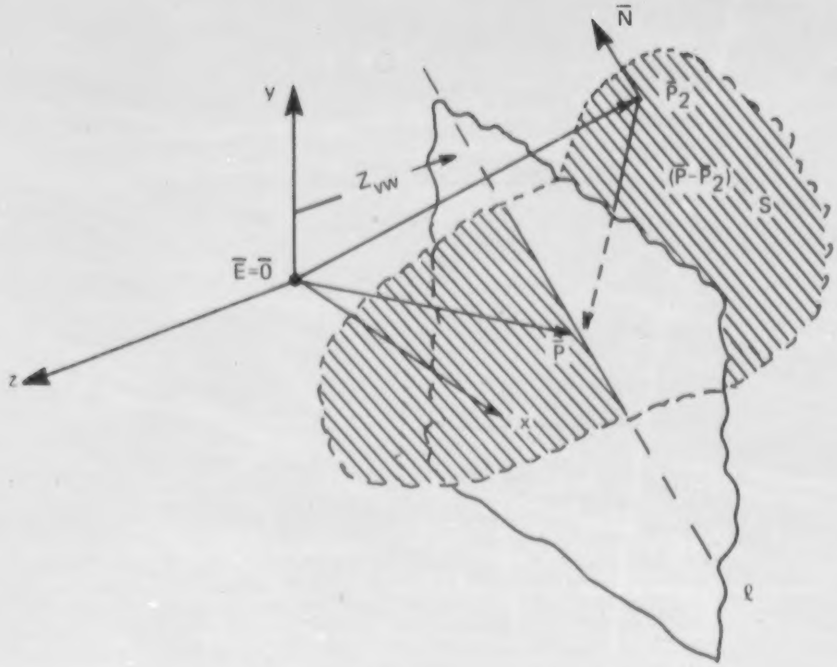


Figure 40 – **Common Point to Both S and $z = Z_{VW}$**

Furthermore, each newly generated edge corresponding to surface S has depth functions $d'_o=1$, $d'_s=0$, and $d'_x=d_x$, where d_x is the incremental horizontal depth function for surface S. Figure 41 shows the data structure of intersecting surfaces and their corresponding y intercepts and edge parameters. Each time an edge that intersects the plane of the viewing window is discovered, a search is made for both the left- and right-side surfaces in the list of surfaces already discovered to intersect the plane of the viewing window. If a surface is found, the new y-intercept is inserted into the y-ordered intercept list for that particular surface. Also, the brightness at that position is calculated using [1.15] and retained with the y-intercept value. If a surface S is not found, then S, d_x, m, and c are added to the surface list and its y-intercept and brightness at the point of intersection with the surface's first entries.

After all the edges of an atom in R^3 have been projected, the surface data table is checked. If the table is empty, the projection processor goes on to the next $([T_i],ATOM_i)$ pair. If the table is not empty, the new edges are created for each intersecting surface passed through the X-Y clipping unit. Only after this has been done does the projection processor proceed to the next $([T_i],ATOM_i)$ pair.

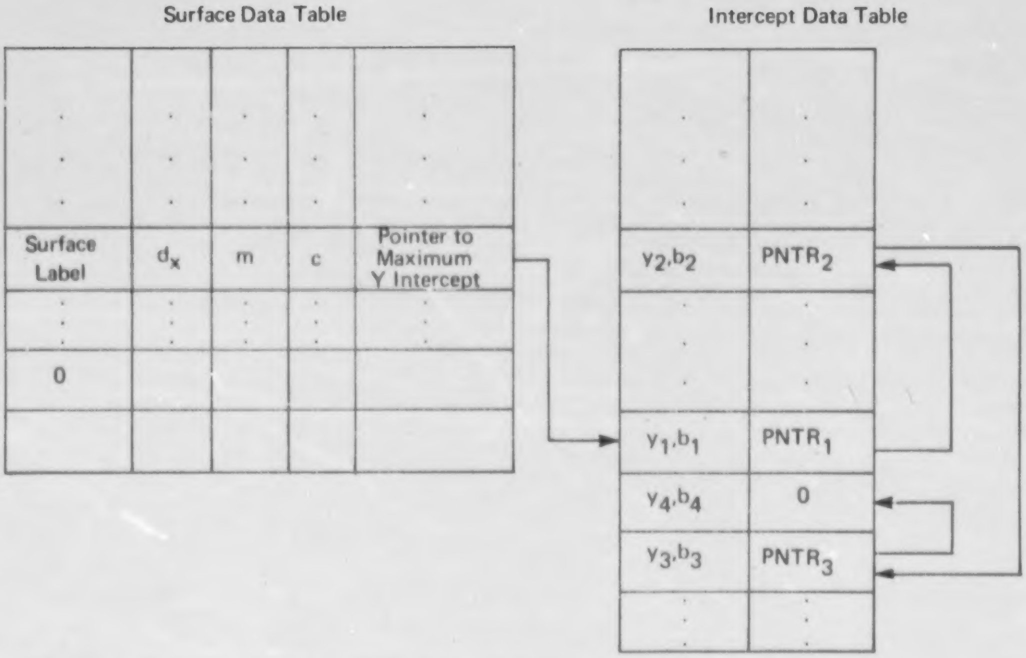


Figure 41 – **Y Intercept Data Structure: $y_1 > y_2 > y_3 > y_4$**

The convention for normal (i.e., far-side) edges is that $\overline{P}_2$ produces a projected point above the projection of $\overline{P}_1$. This defined the left-right convention for the projected surfaces. The convention for edges that intersect the plane of the viewing window is that $\overline{P}_2$ is the end point on the far side of the viewing window. A second algorithm for determining the left-right nature of the surfaces for these edges is now needed. There are now three cases for projected edges:

(1) Horizontal edges: Horizontal projected edges are still deleted, but they generate new surface edge intercepts at the projected y value.

(2) Non-horizontal edges, $d_s < 0$: These edges have $\overline{P}'_2$ as the top projected point as well as $\overline{P}_2$ being the point of the far side of the window. The lower projected point may be calculated by:

$$\Delta y = (1 - d_o)/d_s$$

$$y_1 = y_2 + \Delta y$$

$$x_1 = x_2 + \Delta y \cdot m$$

where d_o is the depth function at $\overline{P}'_2$, the value of the depth function at the plane of the viewing window is +1, and m is the slope of the projected edge.

(3) Non-horizontal edges, $d_s > 0$: These edges have $\overline{P}'_1$ as the actual upper point. Here

$$\begin{aligned} \Delta y &= (1-d_o)/d_s \\ y_2 &= y_1 \\ x_2 &= x_1 \\ y_1 &= y_1 + \Delta y \\ x_1 &= x_1 + \Delta y \cdot m \end{aligned}$$

For this latter case, the left and right surface labels are switched as are (B_1,B_2) and (B_3,B_4), the brightness labels.

The rule for determining whether a projected surface lies on the left or right side of a newly created surface edge depends solely on the d_x function for the surface. If $d_x < 0$, the surface depth gets greater as the surface is scanned from left to right (increasing x). Hence, the new edge is of the form 0:S. If $d_x > 0$, the surface depth decreases as it is scanned from left to right resulting in a new edge of the form S:0.

Formulation of Atom Edge in R^2

A typical projected and clipped (in X, Y, and Z) edge and its relationship to the R_x by R_y discrete resolution grid of the viewing window is shown in Figure 42.

The depth function at the point (X_2,Y_2) is D_o. The incremental depth along the edge is given by D_s and the incremental depth horizontally to the left of the edge is given by D_x. b_1 and b_2 are the ($\log_a$) brightnesses of surface S_ℓ on the left side of the edge and b_3 and b_4 are the ($\log_a$) brightnesses of surface S_r on the right side of the edge. g_ℓ and g_r are the brightness gradients of S_ℓ and S_r along the edge such that

$$\begin{aligned} b_2 &= b_1 + g_\ell (Y_1 - Y_2) \\ b_4 &= b_3 + g_r (Y_1 - Y_2) \end{aligned}$$

The visible surface processor will determine surface depths and, hence, the visible surface only at discrete positions on the resolution raster. This limits the number of times that surface depths need be calculated and compared to a maximum of $R_x \cdot R_y$ times if the depth is to be calculated at every discrete position.

Consider surface S_ℓ as defined by the edge of Figure 42. The first scanline for which the edge becomes valid starting surface S_ℓ to its left is Scanline Y-j. Subsequent scanlines for which the edge starts surface S_ℓ are Y = j+1, j+2, . . . , k-1,k. Since the

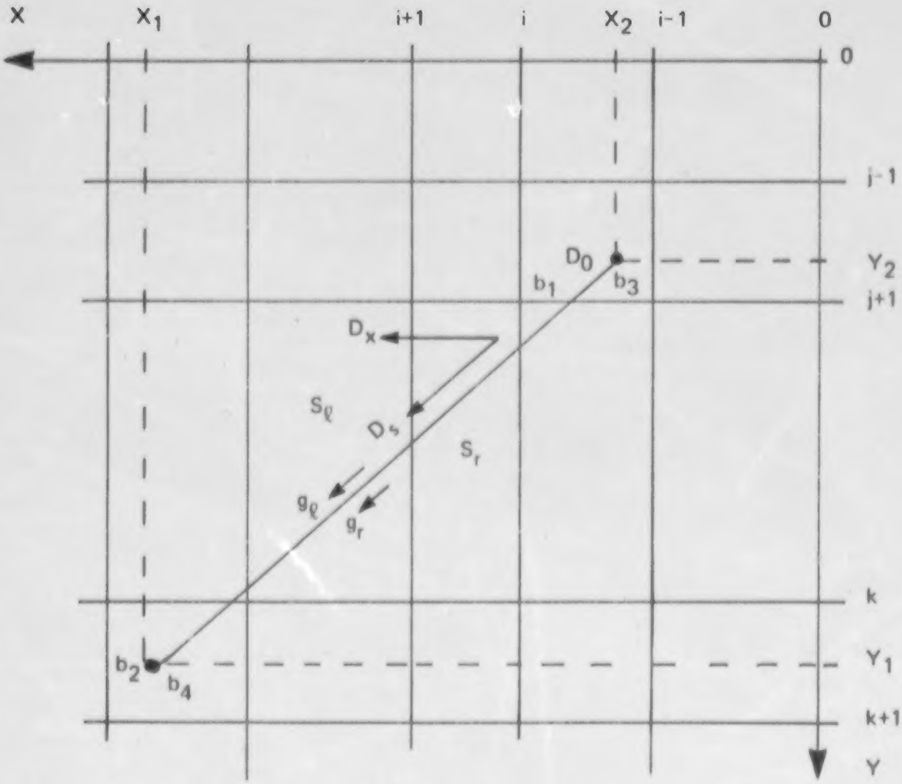


Figure 42 – **Projected Edge on Discrete Raster**

edge starts S_ℓ only over this more limited Y range, the edges are equivalent to a shorter edge that starts exactly on Scanline Y=j and ends exactly on Scanline Y=K. This shorter edge has adjusted parameters given by

$$\begin{aligned} Y_t &= j \\ X_t &= X_2 + M(Y_t - Y_2) \\ D'_o &= D_o + D_s(Y_t - Y_2) \qquad [2.60] \\ b'_1 &= b_1 + g_\ell(Y_t - Y_2) \\ b'_3 &= b_3 + g_r(Y_t - Y_2) \end{aligned}$$

The resulting edge is shown in Figure 43. With the shortened edge of Figure 37, the depth of the surface along the edge at any scanline n scanlines from the top scanline for which the edge is valid is given by $D = D'_o + n\,D_s$. In particular, a rule for updating the depth function D_o from one scanline to the next is now given by

$$D_o = D_o + D_s \qquad [2.61]$$

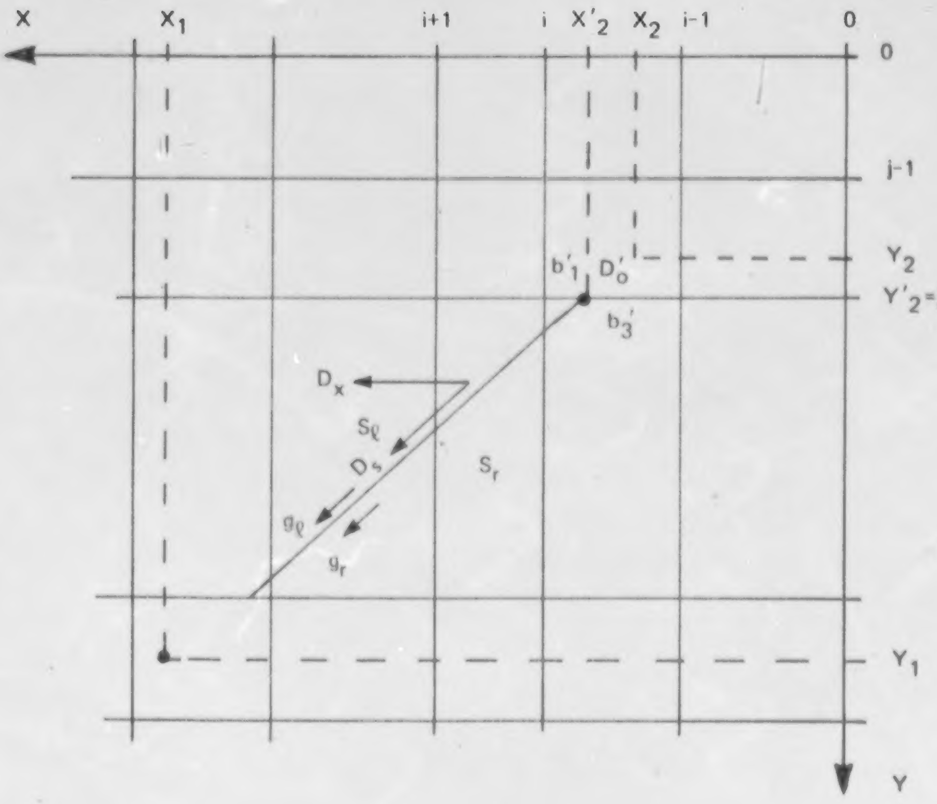


Figure 43 – **Projected Edge Starting and Ending on Discrete Scanlines**

Furthermore, the position of the edge at any scanline n scanlines from the top scanline for which the edge is valid is given by $X = X_t + nM$. In particular, a rule for updating the horizontal edge position X_t from one scanline to the next is now given by

$$X_t = X_t + M \qquad [2.62]$$

The first scanline for which the edge is valid is Scanline j where j is the first integer greater than or equal to Y_2. That is, $j - 1 < Y_2 \leq j$. The last scanline for which the edge is valid is Scanline k where k is the first integer strictly less than Y_1. That is, $k < Y_1 \leq k+1$. This definition assures that a surface will never appear twice on the same line due to two edges terminating at a vertex point that, by chance, is projected exactly onto a scanline. This is demonstrated in Figure 44. This procedure of clipping the tops and bottoms of projected edges exactly at scanlines is easily accomplished by defining a height function H for the edge

$$H = (i_b - i_t) - 1 \qquad i_b - 1 < Y_1 \leq i_b \qquad [2.63]$$

$$i_t - 1 < Y_2 \leq i_t$$

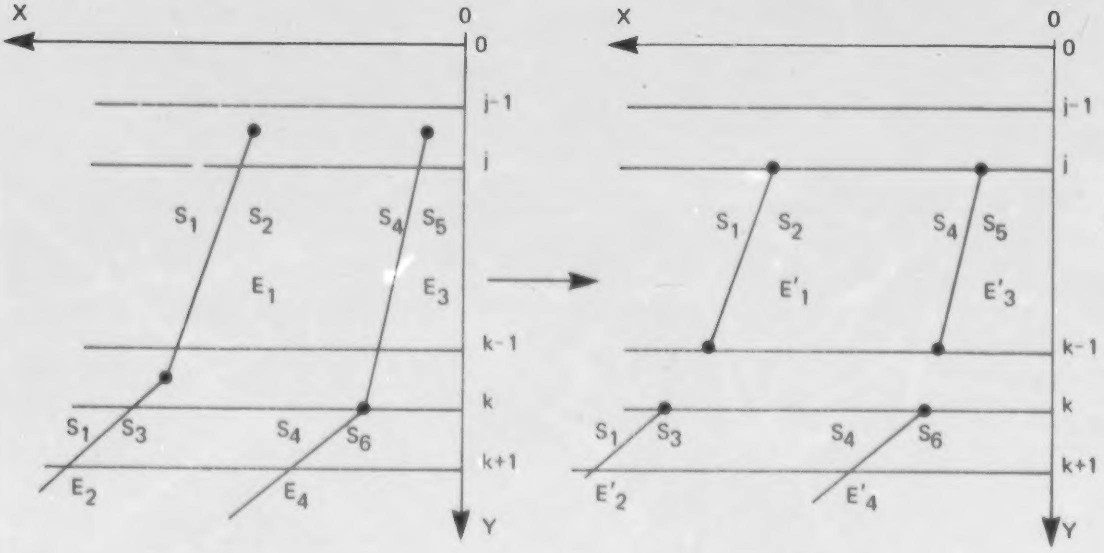


Figure 44 – **Projected Edges Ending on Discrete Scanlines**

If $H \geqslant 0$, edges are clipped as in Figure 44. If $H < 0$, the entire projected edge lies between successive scanlines. Since these edges fail to intersect any scanlines, they are deleted from the projected atom edge set. When an edge first becomes valid at Scanline j_t, its height H is as given by [2.63]. Then, for each successive scanline, the remaining height is given by H-1. So the rule for updating the height from one scanline to the next is given by

$$H = H - 1 \qquad [2.64]$$

H=0 corresponds to Scanline j_b and indicates the last valid scanline for the edge.

The last parameters to be updated from one scanline to the next are the brightness values. These are updated by

$$b_1 = b_1 + g_\ell \qquad [2.65]$$

$$b_3 = b_3 + g_r \qquad [2.66]$$

over the range where $H > 0$.

Thus, once the top of a projected edge is clipped to a Scanline $Y_t = j_t$ via [2.60] and a height function H is defined via [2.63], the edge is fully defined by the set of parameters

$$E_P = (Y_t, H, I_x, X_t, M, S_\ell, S_r, D_o, D_s, D_x, b_1, b_3, g_\ell, g_r, c)$$

where I_x is the valid X indicator function defined by [2.57] for each scanline that the edge crosses. Also to be updated at each valid scanline are H, D_o, X_2, b_1, and b_3. These updates require only addition operations between successive scanlines.

One further modification has been arbitrarily made. Atom surfaces for worlds containing no point light sources are typically flat-shaded. These surfaces have a constant brightness within their edge perimeter. Accordingly, edges with such surfaces on the right side have $b_3=b_4$ and $g_r=0$. These edges are thus used only to end the right-side surface of known brightness. Therefore, the R^2 edge convention is taken to be $(Y_t,H,I_x,X_t,M,S_\ell,-S_r,D_o,D_s,D_x,b_1,g_\ell,c,I_s)$ where I_s is a flag indicating whether or not surface S_ℓ is smooth-shaded or flat-shaded. For cases where S_r is a smooth-shaded surface requiring a brightness function b_2,g_r different from b_1,g_ℓ, a second edge is formed with parameter $(Y_t,H,I_x,X_t,M,-S_\ell,S_r,D_o,D_s,D_x,b_3,g_r,c,I_s)$.

Y-Ordered Output Edge Set

Projected atom edges that survive all the clipping processing are added to an output edge set. This output edge set is ordered according to the Y_t's of the edges. The two tasks involved in adding a projected atom's edges to this set are (a) maintaining the Y-order of the Y_t's and (b) relabeling the surface labels in the projected edges to maintain uniqueness.

The Y-order is maintained through the use of a one-pass bucket sort technique. Two tables are employed. Table Y has R_y positions—one for each of the R_y scanlines. Table E has E_{max} locations, where E_{max} is the maximum number of projected edges allowed. Both tables are initially cleared. As each edge is added to the output set, Table Y's Y_t^{th} position (Y_t is the starting scanline for the edge to be added) is checked for data. If it is null, the edge being added is the first occurrence of an edge with that starting scanline value, and the edge's position q in the output table is placed at position Y_t in Table Y. If position Y_t is not empty but contains some number p, then the p^{th} edge in the output buffer also has Y_t as its starting scanline. Accordingly, the null in the q^{th} position of Table E is replaced with a p and the p in the Y_t^{th} position in Table Y is replaced with a q. After all edges from all the projected atoms have been entered into the output table, Table Y is scanned for data. If the j^{th} position of Table Y contains a k, the k^{th} edge in the output table is read out. Then, position k in Table E is checked for additional edges starting on scanline j. If position k in Table E is null, the reading of Table Y is resumed at position j+1. If position k in Table E is not null but contains ℓ, Edge ℓ in the output table is read out and position ℓ in Table E is checked as before for additional edges starting on scanline j.

Unique surface labeling is achieved as the edges are added to the output edge table. Initially an offset number K is set to one. Then, as each atom's projected edges are added to the output table, +K is added to positive surface labels and -K is added to negative surface labels. 0 labels remain 0. After each atom's edges have been added to the output table, K is incremented by N_S, where N_S is the highest surface label encountered in the atom last processed.* For example, suppose the world consists of two cubes, a cube having six surfaces which are labeled in the atom library 1 through 6. Then the edges for the first cube will enter the output table with surface labels in the range 2 through 7 while the edges for the second cube will enter with surface labels 8 through 13.

Two special edges are always added to the output table for each display. Each has $H=R_y$, $M=0$, $D_o=1$, $D_x=D$, $D_s=0$. The first of these edges has $X_t=0$ and surface labeling 0:1; this edge serves to initialize the display at the right side of the viewing window. The second edge has $X_t=R_x$ and surface labeling 1:0; this edge serves to mark the end of the display at the left side of the viewing window. It is the presence of these edges in every display that initiates the value of the surface label offset number K at one rather than at zero.

Section 3

VISIBLE SURFACE PROCESSOR

INTRODUCTION

This section provides a detailed description of the visible surface processor of the CHARGE image generator. The visible surface processor performs two functions:

(1) It solves the visible surface problem at each of the R_x by R_y resolution points on the observer's viewing window. This process generates the video image of the world as viewed by the observer.

(2) It encodes the video image into a data compressed format, allowing it to be buffered locally at individual CHARGE terminals at a low cost.

Since all displays are locally buffered, the image generator and the communication link to the terminals may be time shared.

INPUT DATA

Input data to the visible surface processor is the Y-ordered output edge set of the projection processor representing potentially visible projected atom surfaces. Each of these projected edges is a 14-tuple of the form

$$E_p = (Y_t, H, I_x, X_t, M, S_\ell, S_r, D_o, D_s, D_x, b, g, c, I_s) \qquad [3.1]$$

where

y_t : First scanline for which E_p is valid
H : Height of projected edge E_p
I_x : Valid X indicator function defined by [2.57]
X_t : Horizontal position at Scanline Y_t
M : Slope of E_p $(M = \Delta X/\Delta Y)$
S_ℓ : Label of left-side surface
S_r : Label of right-side surface
D_o : Value of depth function at (X_t, Y_t)
D_s : Incremental change to depth function along edge
D_x : Incremental change to depth function horizontally
b : log brightness at (X_t, Y_t)

81

g : log brightness gradient along edge

c : log color of S_ℓ

I_s : Indicates if S_ℓ is smooth-shaded

A detailed explanation of these parameters is given in Section 2, Formulation of Atom Edge in R^2 (pg. 72).

OUTPUT DATA

Output data to CHARGE terminals consist of Y-ordered (top to bottom) blocks of X-ordered (left to right) visible edges. Each of these visible edges is an 8-tuple of the form

$$E_v = (Y_t, H, c, X_t, M, b, g, I_s) \qquad [3.2]$$

where

Y_t : First scanline for which E_v is valid

H : Height of visible edge E_v

c : log of color to right of visible edge E_v

X_t : Horizontal position of E_v on Scanline Y_t

M : Slope of E_v ($M = \Delta X/\Delta Y$)

b : log of brightness at (X_t, Y_t)

g : log of brightness gradient along E_v

I_s : Indicates if right side is to be smooth-shaded

Each such visible edge controls the color and the brightness of an area to its right; hence, the X-ordered set of edges within a block B_i controls that area of the TV screen for which block B_i is valid. The first edge within a block is a specially encoded "marker edge" which serves to both identify the start of the block and to set the Y-limits over which the block is valid. Furthermore, special "null edges" may be located anywhere within the block. The number of edges within a block is bounded by

$$N_B = N \cdot (256 - [\Delta Y/40]) \qquad [3.3]$$

where

N : Number of parallel memory banks assigned to the CHARGE terminal.

ΔY : Y-range of the block

$[\Delta Y/40]$: Truncated integer

A detailed description of the coding of the eight visible edge parameters into an 76-bit word and the CHARGE terminal memory structure is found in a separate report.[1]

---

[1] Ronald J. Swallow, *CHARGE Interactive Graphics System Terminal: Theory of Operations*, HumRRO Technical Report 74-26, December 1974, 74 pp.

OVERVIEW OF SCANLINE OPERATION

The operation of the visible surface processor is on a successive scanline basis whereby each scanline represents a discrete Y resolution value of the observer's viewing window (Figure 32). This operation is as follows:

(1) Starting at the top of the observer's viewing window (Y=1), the visible surface processor partitions each scanline in the X interval $[R_x,1]$ into a set of mutually exclusive, collectively exhaustive, subintervals of the form $(X_{i+1},X_i]$ such that either no surface or exactly one surface is visible in each subinterval. This is accomplished by determining the nearest surface for all discrete values of X in the interval $[R_x,1]$.

(2) As it partitions successive scanlines, the visible surface processor creates and maintains an X-ordered set of straight line visible edges that affect the same scanline partitioning as in (1). As the scanline partitioning changes from one scanline to the next, some edges in this set remain valid and may be extended, other edges cease to be valid and are deleted from the current set, and still other visible edges must be created and added to the current visible edge set to realize the current scanline partitioning. Visible edges deleted from the current visible edge set remain as X-ordered members of the output visible edge block whose size is bounded by [3.3].

(3) Steps (1) and (2) are repeated until $Y=R_y$. At this point, the final block of visible edges is complete and additional blocks filled with marker edges and null edges are created to fill up the CHARGE terminal's memory banks and to provide null data for those scanlines corresponding to vertical retrace time of the CHARGE terminal's monitor.

Figure 45 demonstrates for a single Scanline Y=i the data flow from the input data set of Y-ordered projected edges to the output block of X-ordered visible edges. At the start of Scanline Y=i, any edges in the input data set that become valid at Y=i are added to the set of projected edges that were added on preceding scanlines but which remain valid (due to their height values, H). Companion to this current projected edge buffer is a pointer list that maintains a right to left X-ordering of the projected edge set at the resolution of R_x discrete X intervals per scanline. Each edge in the current projected edge set points to a next edge whose X value lies either within the same X resolution interval or in a higher X resolution interval.

The visible surface algorithm processes these edges in X order from right to left, generating horizontal intervals over which either no surface or the same surface is visible. Each edge as it is encountered tries to end output attempts for the surface on its right side and to start output attempts for the surface on its left side. The visible surface

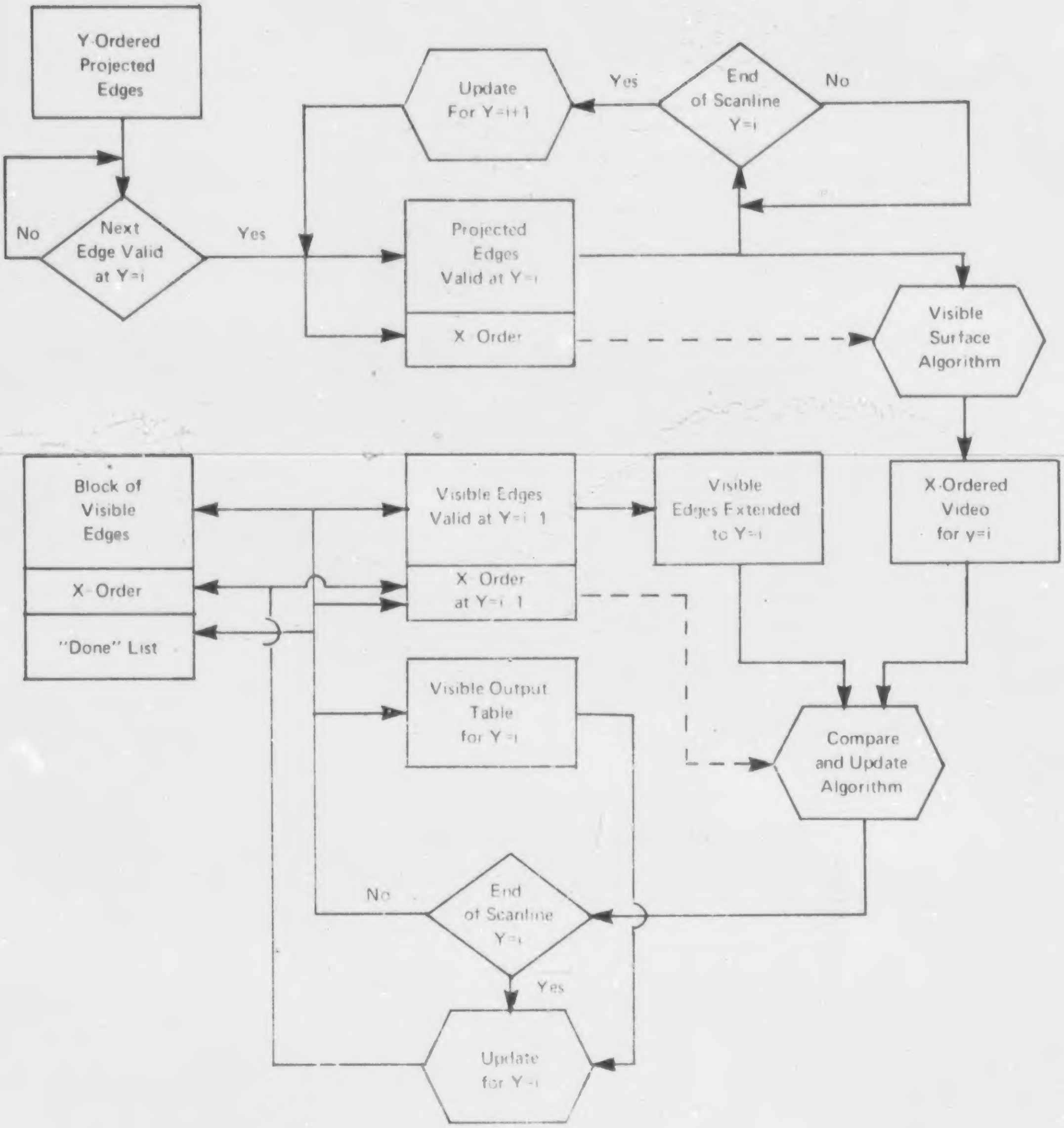

Y Ordered Projected Edges
No
Next Edge Valid at Y=i
Yes
Update For Y=i+1
Yes
End of Scanline Y=i
No
Projected Edges Valid at Y=i
X Order
Visible Surface Algorithm
Block of Visible Edges
X Order
"Done" List
Visible Edges Valid at Y=i 1
X Order at Y=i 1
Visible Edges Extended to Y=i
X Ordered Video for y=i
Visible Output Table for Y=i
Compare and Update Algorithm
No
End of Scanline Y=i
Yes
Update for Y=i

Figure 45 – **Visible Surface Processor Functional Overview**

84

algorithm utilizes the surface depth parameters included in the projected edge definition to determine that surface whose distance from the observer's eye is minimum. When the minimum depth surface changes, the visible surface algorithm forms a partition between the two minimum surface intervals; the attributes that it assigns to the partitioning boundary are those that are necessary to uniquely match a visible edge partition.

At the end of the scanline ($X=R_X$) the partitioning is complete and the set of current projected edges may be updated for the next scanline ($Y=i+1$). This updating process for each edge involves:

(1) $H=H-1$. If $H<0$, the edge is not valid for $Y=i+1$ and may be deleted from the current valid projected edge set.

(2) $D_O=D_O+D_S$. This updates the depth function at the top of the edge.

(3) $X=X+M$. This updates the X position of the top of the edge.

(4) I_X, the X indicator function as defined by [2.57] is updated by the new X value.

(5) The X-ordered list is updated by the I_X value, if necessary. This crossover of edges is a relatively low frequency event and does not require a complete X-ordering of the entire current projected edge set.

In essence, the update procedure "shortens" each edge by clipping off the top of an edge and redefining the new top in X,Y, and depth.

Analogous to the set of current projected edges at $Y=i$ and its companion X-order list are the set of current visible edges at $Y=i-1$. When the actual video scanline partitioning has been completed, an attempt is made to match each successive actual video partition with a visible edge extended to scanline i. The order of this matching attempt is governed by the X-ordering of the preceding scanline's visible edges. As visible edges are matched, they are deleted from the $Y=j-1$ ordering list.

If a matched visible edge is not the first edge in the remaining $Y=i-1$ order list, those edges preceding it in the list are deemed not to be extendable onto scanline $Y=i$ (since visible edges may never cross) and are ended at scanline $Y=i-1$. If an actual scanline partition has no corresponding visible edge, it is deemed to be "new" and a visible edge is started for it. This new or matched visible edge is next compared in X with the preceding edge to determine if it is "too close." Visible edges surviving this comparison generate a visible output table for scanline $Y=i$. At the end of the scanline, this table is used to generate an X-order list for the current visible edge data set and to insert into proper X-order any new visible edges added to the visible edge block.

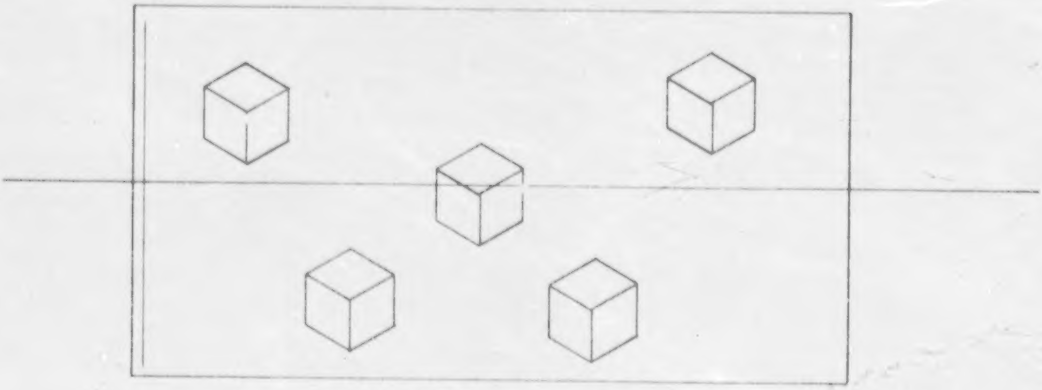
The visible edge block is ended and sent (read out according to its X-order list) whenever the number of edges in it plus the number of actual video partitions exceeds N_B as defined by [3.3]. This guarantees that each scanline is uniquely defined by only one visible edge block. As a visible edge block is read out, each of its positions except for those corresponding to edges that still appear in the current visible edge set is freed up. Hence, the initial X-ordering for the next visible edge block is that of those edges extending down from the preceding edge block. Thus, if the image is as in Figure 46 and blocking occurs at $Y=Y_m$, the corresponding visible edge blocks are those of Figure 47 and 48. Note that while edges do not extend down into subsequent blocks, any edge may have its origin defined in the Y-range of any preceding block.

VISIBLE SURFACE ALGORITHM

Scanline operation of the visible surface algorithm is shown in Figure 49. Initially, the output table, the pending delayed remove table, the change table, and the pending brightness table are cleared. X, the horizontal position register, is set to zero and the first "beam box" is loaded with surface "1" and the others are cleared. This first beam box will always be cleared by the 0:1 edge at X=1 that is present in every image to turn on the appropriate background color and brightness (see Section 2, y-Ordered Output Edge Set, pg. 77). Also initialized at the start of each scanline are 12 memory registers that are used to determine for each X whether or not a new video partition should be added to the output table:

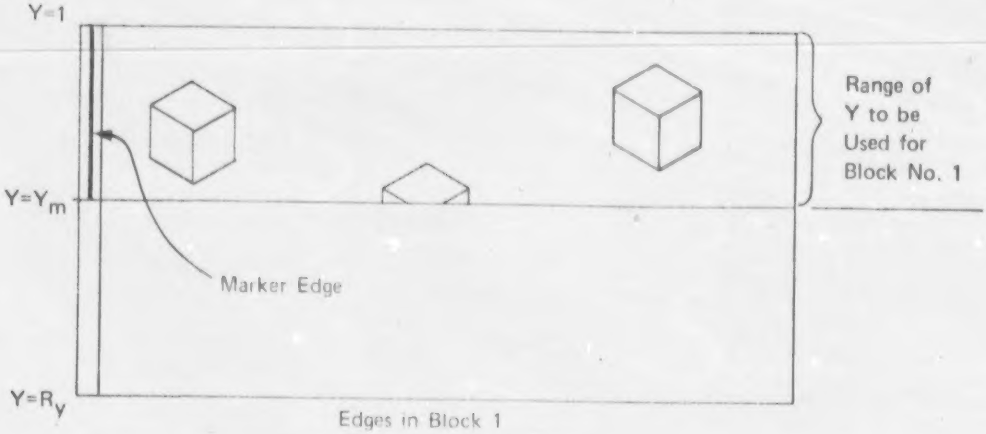
(1) ICNAM : Surface with minimum depth at $X=i$
(2) ILNAM : Surface with minimum depth at $X=i-1$
(3) ICPNT : Beam box containing ICNAM
(4) ILPNT : Beam box containing ILNAM
(5) DC : Depth of ICNAM at $X=i$
(6) DL : Depth of ILNAM at $X=i-1$
(7) ICSM : Smooth-shading bit for ICNAM
(8) ILSM : Smooth-shading bit for ILNAM
(9) ICC : Color of ICNAM
(10) ILC : Color of ILNAM
(11) BC : Brightness of ICNAM
(12) BL : Brightness of ILNAM

At the start of each scanline ICNAM=1, ICPNT=1, DC=1, and ICSM=0.



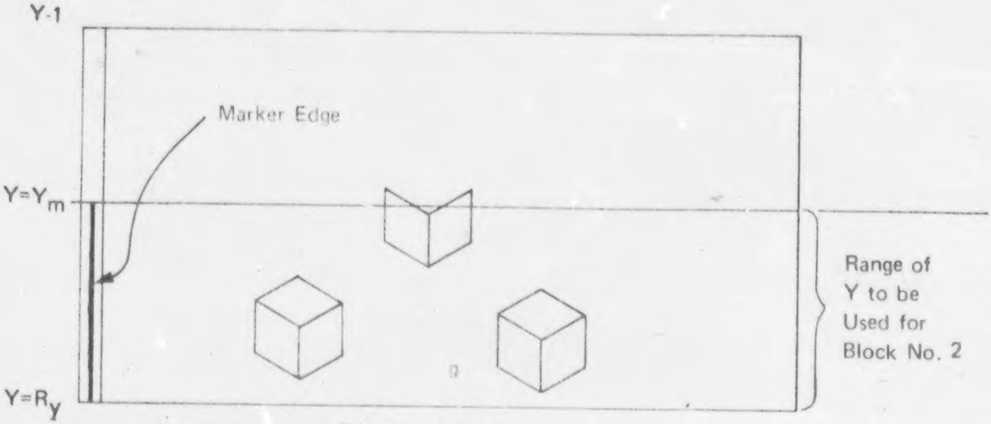
Edges Defining Image

Figure 46 – **Edges Defining Image**



Edges in Block 1

Figure 47 – **Edges in Block 1**



Edges in Block 2

Figure 48 – **Edges in Block 2**

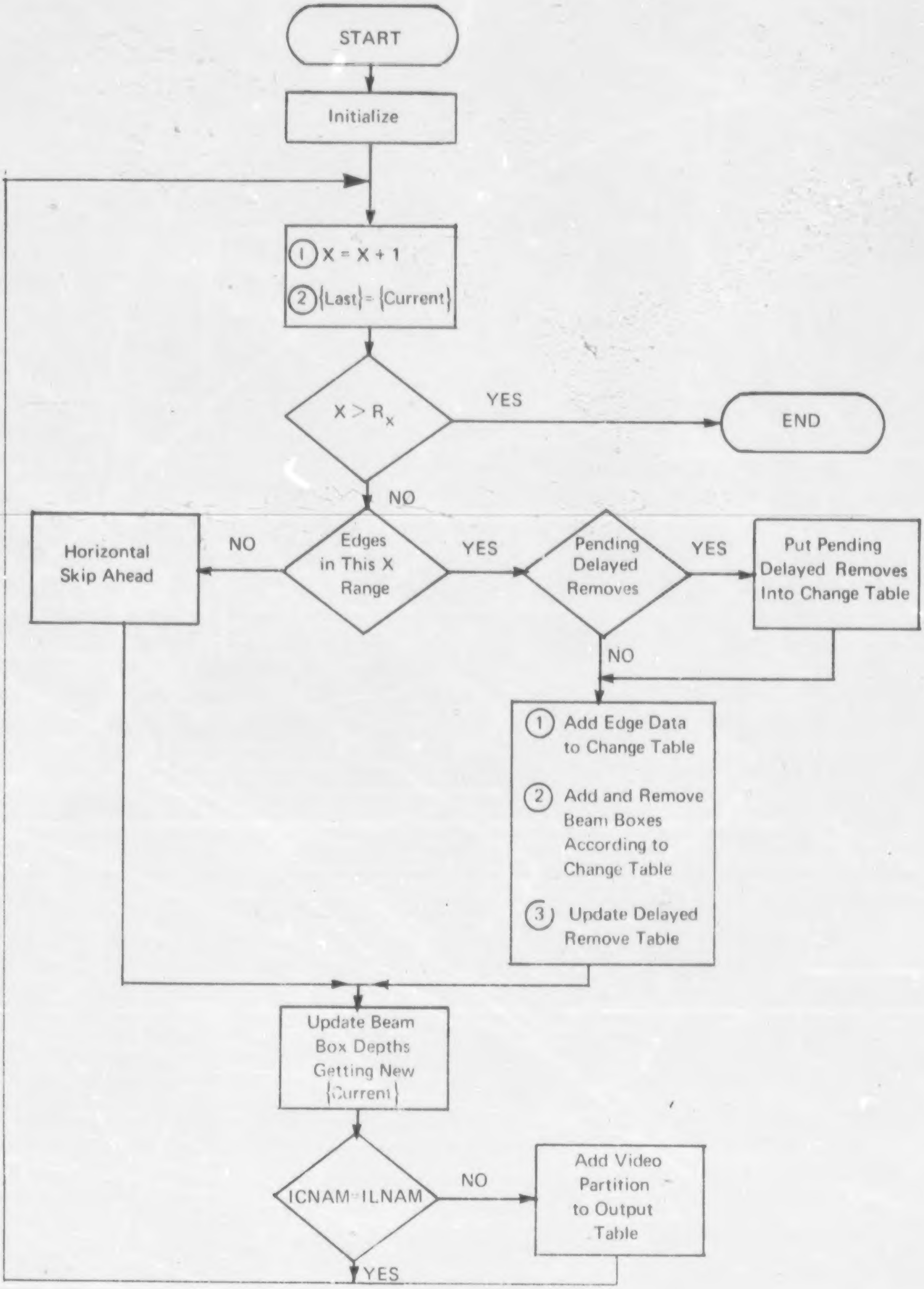

START
Initialize
① X = X + 1
② {Last} = {Current}
X > R_x
YES
END
NO
Horizontal Skip Ahead
NO
Edges in This X Range
YES
Pending Delayed Removes
YES
Put Pending Delayed Removes Into Change Table
NO
① Add Edge Data to Change Table
② Add and Remove Beam Boxes According to Change Table
③ Update Delayed Remove Table
Update Beam Box Depths Getting New {Current}
ICNAM=ILNAM
NO
Add Video Partition to Output Table
YES

Figure 49 – **Visible Surface Algorithm**

After initialization, a processing loop transforms the input set of X-ordered projected edges into an X-ordered output set of visible surface partitions (the visible surface between partitions is that surface whose depth is minimum over that X interval between partitions). This processing loop operates only at the discrete values $X=i\in\{1,2,\ldots,R_x\}$, and the operation is in one of two modes depending upon whether or not there are projected edges in the interval $[i,i-1)$. If edges are present in that interval, those beam boxes containing the edges' right-side surfaces are unloaded and reloaded with the label and depth information of the edges' left-side surface; all of the beam boxes are then updated to $X=i+1$ and the surface with minimum depth at $X=i+1$ is determined. If the next edges are present at $X=j$ $(j>i)$, all of the surfaces that are potentially visible at $X=i+1$ are potentially visible at $X=j-1$; hence, a horizontal skip-ahead attempt may be made by updating each beam box by $(j-i-1)$ units. If the same surface visible at $X=i$ is visible at $X=j-1$, the skip-ahead is known to be valid since all surfaces are planar.

The format of a beam box is shown in Figure 50. The depth of the surface in the perspective domain is given by parameter $D(1\geq D>0)$. D_x is the incremental change in the perspective depth in the horizontal direction and DSAVE is a register used to save the value of D during a horizontal skip-ahead attempt; D is restored from DSAVE should the skip-ahead attempt prove unsuccessful. LABEL identifies the surface, ISM is a bit that is turned on if the surface is smooth-shaded, and PNTR points to the edge in the X-ordered current projected edge set that was used to turn on the beam box. The number of beam boxes loaded with surfaces at $X=i$ is the number of overlapping surfaces at $X=i$.

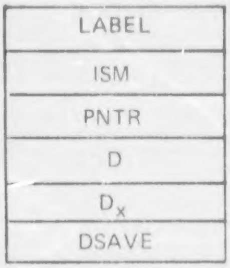


Figure 50 – Beam Box Format

Each beam box is loaded in such a way that its depth parameter D represents the surface perspective depth precisely at the discrete values of X along the scanline. This assures that all surface depth parameters will be consistently compared. Consider Figure 51. Edge E crosses the scanline j at $i\geq X_t>i-1$, causing the edge to be encountered in the i^{th}

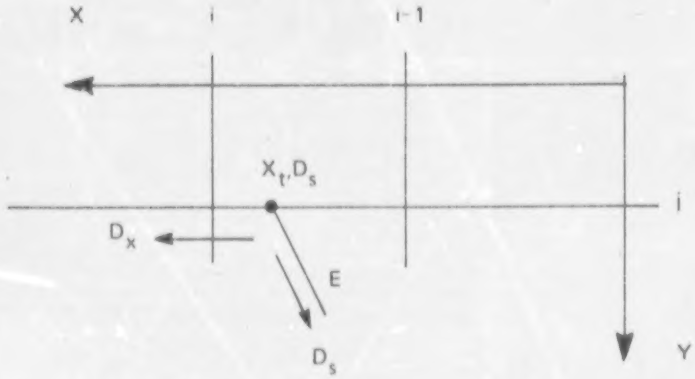


Figure 51 – **Edge E in ith X Range**

X range. The depth function D is to be loaded such that it will be a correct measure of the left surface depth at X=i,i+1, . . . after updating. This is easily accomplished by setting

$$
\begin{aligned}
D &= D_0+(i-X_t)D_x-D_x \\
&= D_0+(i-X_t-1)D_x \qquad [3.4]
\end{aligned}
$$

That is, the beam box is loaded as if the left-side surface existed at X=i-1; then, as the beam boxes are incremented forward 1 unit by adding D_X to D, all the boxes refer to depths at X=i.

Often more than one edge may reference the same surface within an X resolution interval. This situation occurs when edges converge at a vertex point of an atom—particularly if the projected atom size is small and of the same order of magnitude as the rastering (W_X/R_X). To handle cases such as these, all of the edges within the same X interval are scanned before any changes are made to the beam boxes. This scanning procedure makes entries in the Change Table; each entry in the Change Table has the form shown in Figure 52.

LABEL
Net Change
PNTR1
PNTR2

Figure 52 – **Change Table Entry**

As each surface is encountered in edge definitions, a running count is kept of the number of times that it appeared on the left side of an edge (add) minus the number of times that it appeared on the right side of an edge (remove). Then

$$\text{Net change} = \#\text{ adds} - \#\text{ removes} \qquad [3.5]$$

If net change is positive, a beam box for the surface is added (if one does not already exist). If net change is negative, the corresponding surface beam box is removed (if it exists). If net change is zero, no attempt is made to either add or remove a beam box for that surface. If an attempt is made to remove a beam box that does not exist, that attempt will be remembered in a "delayed remove" table and will be used to negate any single attempt at a successive X location to add a beam box for that surface. These delayed removes are caused by slightly inaccurate (usually very large) projected edge slopes as in Figure 53. PNTR1 points to the edge (at X=i) in the current projected edge set with LABEL on the left side and (a) ILNAM or (b) minimum surface label on the right side. PNTR2 points to the edge (at X=i) in the current projected edge set with LABEL on the right side and a minimum surface label on the left side. Usually either PNTR1 is zero (remove only) or PNTR2 is zero (add only).

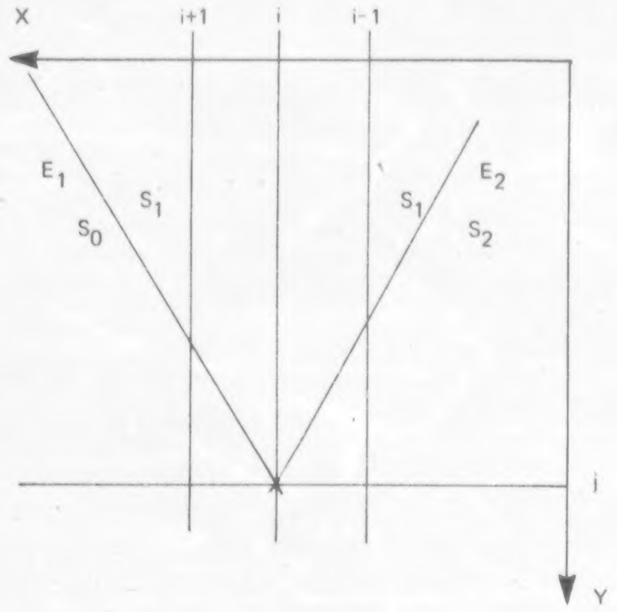


Figure 53 – **Delayed Removal of Surface S_1 at X = i + 1 on Scanline j**

Once all of the changes have been made to the beam boxes and the beam boxes have been updated by one X resolution level, a comparison of the depth parameters is made to determine which beam box contains the surface with minimum depth. If two surfaces have the same minimum value of D, selection of the surface with the minimum surface label guarantees consistency in resolving such conflicts. Selection of a beam box sets ICNAM, ICPNT, and DC. The pointer contained in the beam box is then used to obtain values for ICSM, ICC, and BC.

At this point, the test for change in output is made: ICNAM and ILNAM are compared; if ICNAM=ILNAM, the surface visible at X=i-1 remains visible and no changes to the output set are required. If, however, ICNAM≠ILNAM, a change in visible surface has occurred in the ith X interval and a new partitition must be added to the output video partition table. The partition between ICNAM and ILNAM is a straight line edge in the plane of the viewing window and may be caused by one of three situations:

(i) ICNAM may have been added by an edge of the form ICNAM:S in the i^{th} X range, where S is any surface label. Typically, however, either S=ILNAM corresponding to an edge between surfaces on the nearest atom, or S is negative corresponding to the rightmost edge of a new atom that lies closer to the eye than that atom containing surface ILNAM.

(ii) ILNAM may have been removed by an edge of the form S:ILNAM in the i^{th} X range, where S is any surface label. Typically, either S=ICNAM corresponding to an edge between surfaces on the nearest atom, or S is negative corresponding to the leftmost edge of the last visible atom that now ceases to be visible.

(iii) There may be an intersection between ICNAM and ILNAM.

In Figure 54, scanline j is partitioned into seven regions by three surfaces. S_2 lies in front of S_1, and S_3 intersects S_1 in a line passing through point c. Each partition is also labeled as to being caused by Cases (i), (ii), or (iii).

The partitioning is:

(1) (a,1] , ICNAM=0
(2) (b,a] , ICNAM=S_1
(3) (c,b] , ICNAM=S_3
(4) (d,c] , ICNAM=S_1
(5) (e,d] , ICNAM=S_2
(6) (f,e] , ICNAM=S_1
(7) [R_x,f] , ICNAM=0

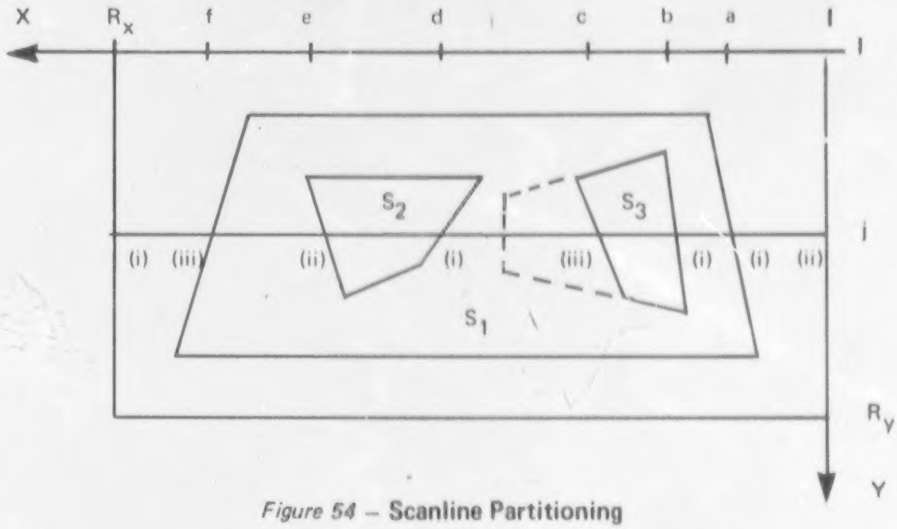


Figure 54 – **Scanline Partitioning**

where 0 denotes "background only"; that is, no projected atom is visible and the video is the background color and brightness. The partitions at X=1 and $X=R_x$ are Cases (ii) and (i), and are due to the 0:1 and 1:0 edges present in all displays.

The format for a video partition is shown in Figure 55. This format contains all of the information required to match an extended visible edge or to start a visible edge should the partition fail to match an existing visible edge. S_ℓ and S_r are left and right surface labels for the partition. S_{last} is set to ILNAM and COLOR=ILC, thereby referring to the surface to the *right* of the partition. TYPE=2·ICSM+ILSM, thereby indicating which of the following smooth shading transitions has occurred:

TYPE = 0 = 00 → flat : flat
1 = 01 → flat : smooth
2 = 10 → smooth : flat
3 = 11 → smooth : smooth

If the partition is due to a projected atom edge as in Case (i) or (ii), one or both of the NEW/LAST bits may be set. The NEW bit is set if the edge in the current projected edge set has become valid for the first time; that is, the edge has just joined the current projected edge set. The NEW bit set causes the partition to generate a new visible edge since any match with an existing visible edge would obviously be erroneous. The LAST bit is set if the edge in the current projected edge set has H=0; that is, the edge will be deleted from the current projected edge set at the conclusion of the present scanline. The LAST bit set causes its corresponding visible edge (matched or new) to be terminated on

S_ℓ
S_r
S_{last}
COLOR
TYPE
NEW/LAST
X_p
M_p
b_p

Figure 55 – **Video Partition Format**

the present scanline since any extension downward to succeeding scanlines would obviously be erroneous. X_p,M_p, and B_p reflect the position, slope, and brightness (to the right) of the video partition.

All of the data required for a video partition are available at the instant it is created except, perhaps, for the value of b_p. If the partition is due to a projected atom edge, then S_ℓ,S_r, NEW/LAST, and M_p are obtained from that edge while S_{last}, COLOR, TYPE, and X_p are obtained from ILNAM, ILC, ICSM, ILSM, and X. If the partition is due to an intersection, then S_ℓ=ICNAM, S_r=ILNAM, NEW/LAST=00, and M_p is calculated from the edge data of the edges pointed at by beam box #ILPNT and beam box #ICPNT. Consider Figure 56. Edge E1 with Slope M_2 loads a beam box with S1 and Edge E2 with Slope M_2 loads a beam box with S2. At X=i, ICNAM=S1 and ILNAM=S2. At X=X_p on Scanline j, both S1 and S2 have a perspective depth of D=D′. The dashed line of intersection between S1 and S2 represents the locus of points on S1 and S2 having the same perspective depth.

In particular, at Y=j+1

$$D(S_1) = D(S_2)$$

$$D'+D_{y_1}+M_p \cdot D_{x_1} = D'+D_{y_2}+M_p \cdot D_{x_2}$$

$$D_{y_1}+M_p \cdot D_{x_1} = D_{y_2}+M_p \cdot D_{x_2}$$

But, since S_1 and S_2 are planar, $D_{s_i} = D_{y_i}+M_i \cdot D_{x_i}$ and so

$$D_{s_1}-M_1 \cdot D_{x_1}+M_p \cdot D_{x_1} = D_{s_2}-M_2 \cdot D_{x_2}+M_p \cdot D_{x_2}$$

or

$$M_p = \frac{(D_{s_2}-M_2 \cdot D_{x_2})-(D_{s_1}-M_1 \cdot D_{x_1})}{(D_{x_1}-D_{x_2})} \qquad [3.6]$$

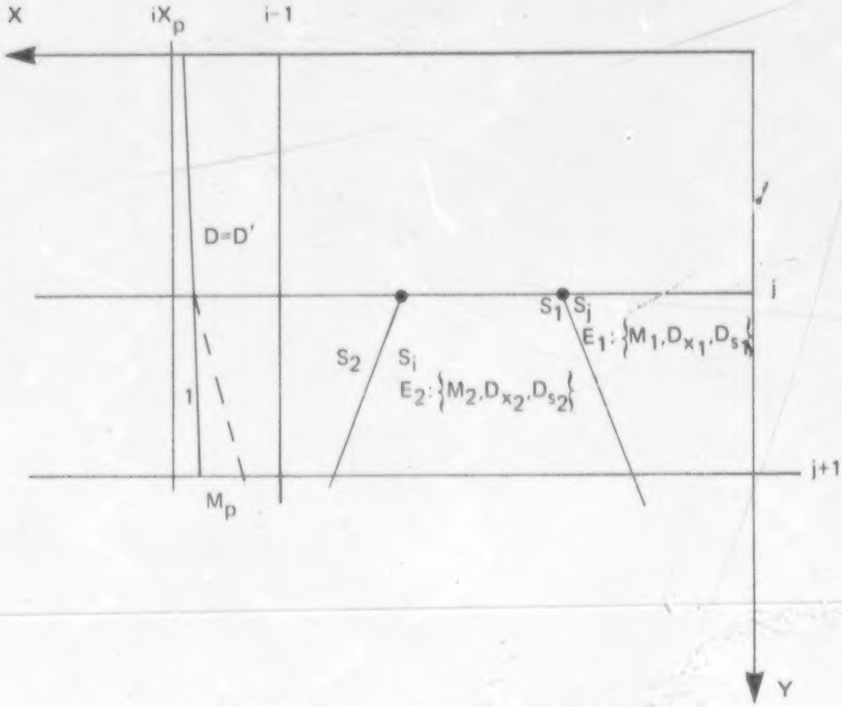

X
iX_p
i-1
D=D'
j
S_1
S_j
$E_1: \{M_1, D_{x_1}, D_{s_1}\}$
S_2
S_i
$E_2: \{M_2, D_{x_2}, D_{s_2}\}$
1
j+1
M_p
Y

Figure 56 – **Intersection of Visible Surfaces**

If surface ILNAM is not a smooth-shaded surface, b_p=BL since the brightness at any point on a flat-shaded surface is that of the surface when the beam box was loaded by a right-side edge. If ILNAM is a smooth-shaded surface, and there exists an edge of the form S:ILNAM within the interval (i,i-1] where $i \neq R_x$, then the brightness at $X=X_p$ is given by the brightness of the edge; if, however, $i=R_x$, then the edges' horizontal position discretely limited to R_x by the indicator function I_x (given by 2.57) for X-ordering reasons, may not be R_x in which case b_p is given by

$$b_p = b_r + \frac{(X_p - X_r)}{(X_\ell - X_r)} (b_\ell - b_r) = b_r + (X_p - X_r)g \qquad [3.7]$$

where the subscripts ℓ and r designate the left and right edge parameters and X_ℓ and X_r may lie anywhere on the scanline subject only to $X_\ell \geq X_r$.

If no edge of the form S:ILNAM exists within the i^{th} X interval, the value of b_p cannot yet be ascertained. In these cases, the brightness is deemed to be pending and an entry is made in the "pending brightness table," which has the form shown in Figure 57 for surface S_i with three pending brightnesses at positions 0_1, 0_2, and 0_3 in the output video partition table. When an edge of the form $S{:}S_i$ is encountered yielding a remove beam box S_i, the S_i of the edge is matched against the S_i in the pending brightness table indicating that one or more pending brightnesses must be calculated. The horizontal brightness gradient is first calculated using the brightness and X position data of the two edges that load and unload the beam box

$$g = \frac{b_\ell - b_r}{X_\ell - X_r} \qquad [3.8]$$

Once this gradient has been calculated, each of the b_p's in the pending brightness table may be calculated by [3.7]. Referring back to surface S1 in Figure 54, the brightnesses to the right for the partitions at X=b and X=d are calculated at X=f if surface S_1 is a smooth-shaded surface.

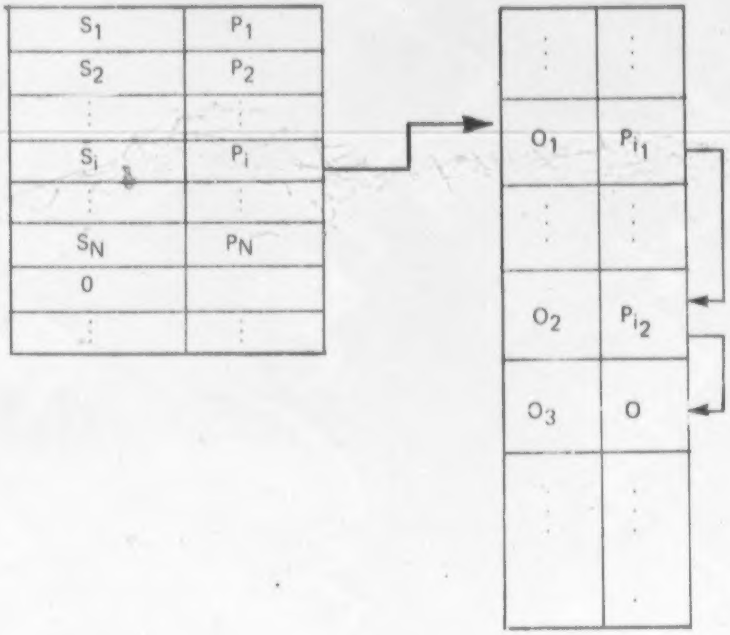


Figure 57 – **Pending Brightness Table Form**

If ICNAM is a smooth-shaded left-side surface, and there is no edge of the form ICNAM:ILNAM in the i^{th} X interval, the image generator must form two partitions. The first partition is at $X=X_p$ and affects the desired right-side brightness and color. Furthermore, TYPE=ILSM controls smooth shading across the right-side surface. The second partition is formed at $X=X_p+2$ (minimum separation for the CHARGE terminal); this second edge has TYPE=2, COLOR=ICC, and is used to provide a brightness for the CHARGE terminal to smooth-shade toward as it decodes the left-side surface.

The addition of this edge at $X=X_p+2$ is always tentative and will be deleted should a partition occur within $[X_p+2,X_p)$ with a right-side surface S_r which is not equal to the left-side surface S_ℓ of the double partition at X_p. If $S_r=S_\ell$, the new partition is deleted and the old partition remains valid. Figures 58 and 59 demonstrate both of these cases. Surfaces S_0, S_1, and S_2 are surfaces on a cylinder atom; S_0 forms one of the flat cylinder ends while S_1 and S_2 are two smoothly shaded facets of the cylinders round outer surface. Surface S_3 lies on a plane nearer to the eye of the observer than the cylinder.

On Scanline i, a double edge is created to account for the smooth-to-flat $S_1:S_0$ visible surface switch. Farther down the scanline there is a $S_2:S_1$ partition (no brightness discontinuity), and still farther down the scanline is a $S_3:S_2$ partition. On scanline i+1, the double partition is initially formed to affect the $S_1:S_0$ visible surface switch; however, before X can increment by 2, the $S_2:S_1$ visible surface switch occurs. Since surface S_1 appears in both places, this second partition may be safely deleted since the brightness function defined over the curved cylinder is continuous.

At Scanline k, however, the $S_3:-S_4$ (S_4 is the back surface of the plane) partition occurs within two X resolution intervals of the $S_2:S_0$ partition. In this case, to allow the TYPE=2 $S_2:S_0$ partition to survive would be to erroneously move it across a brightness/color discontinuity. Figure 59 demonstrates the final correct video partitioning of this region of interest with arrows indicating smooth shading. Note that on scanline k the $S_3:-S_4$ edge incorrectly shades toward the brightness of S_0, the flat end of the cylinder, rather than toward S_1, the rounded surface.

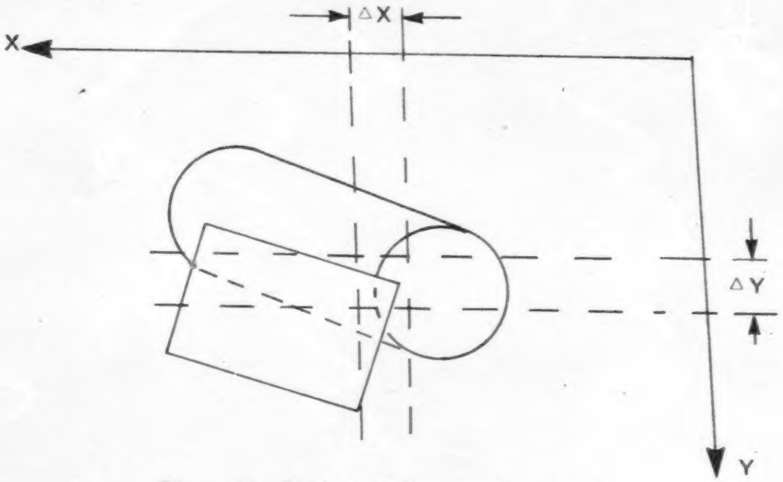


Figure 58 – **Projected Plane and Cylinder**

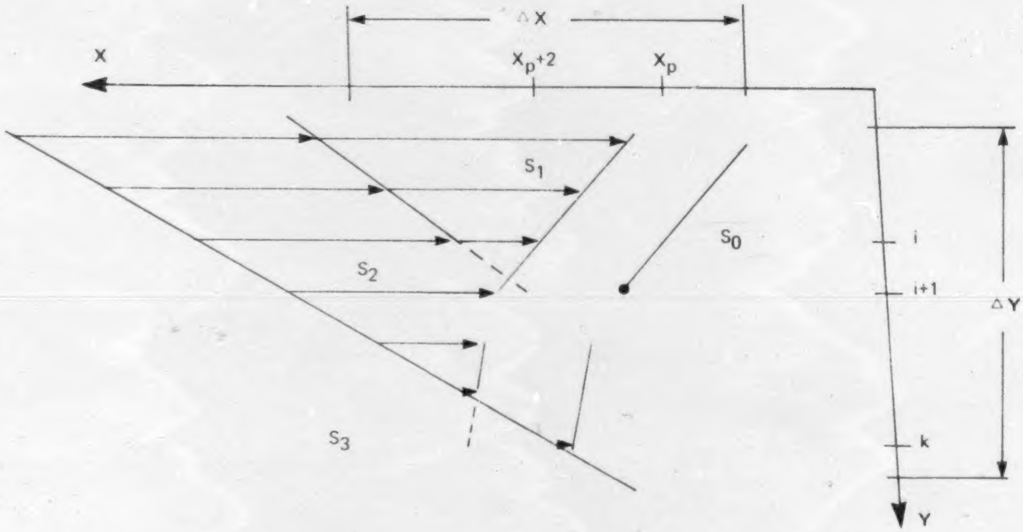


Figure 59 – **Close-up View of Plane and Cylinder**

This error will be unnoticed by the eye since either:

(i) The distance between the two partitions is less than two X resolution intervals (out of 1600).

(ii) The S_3:-S_4 partition may be shortened to scanline k-1 by the matching algorithm, thereby causing flat surface S_3 to "jut" over by this small distance to S_0 on scanline k.

If a video surface switch is due to an edge, as in Case (i) or (ii), care must be taken to select the correct edge. Consider, for example, Figure 60. Surfaces S_1,S_2, and S_3 all lie on the same atom. At X=i, there is a visible surface switching from S_1 to S_3. Within the i^{th} X interval are two edges: E_1 of the form S_2:S_1, and E_2 of the form S_3:S_2. Edge E_1 is Case (ii), Edge E_2 is Case (i). The problem is to select the parameters of one of the edges for the partition.

A second ambiguous case is that of Figure 61 in which several edges reference S_1 as the left-side surface (and S_2 as a right-side surface).

A third case is that of Figure 62. In the first case, the new visible surface, S_1, actually lies in front of S_2 while in the second case, S_2 is in front of S_1.

These ambiguous situations may be consistently resolved by the processing steps of Figure 63 which utilize ICNAM, ILNAM, DC, and DL. NBEAM contains the number of beam boxes vying for output.

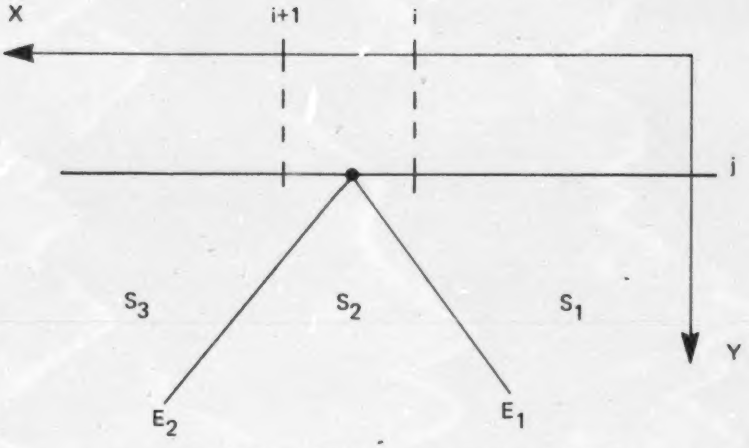


Figure 60 – **S_3:S_1 Ambiguous Scanline Partition**

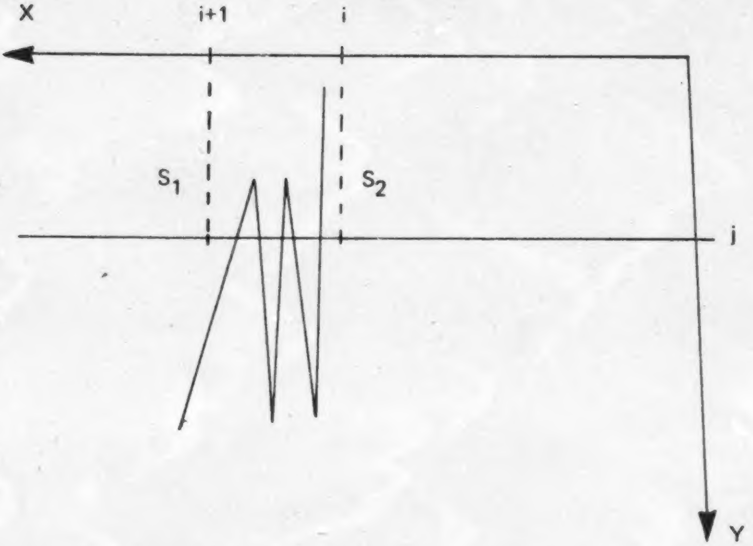


Figure 61 – **Multiple Edges Referencing S_1 on Left**

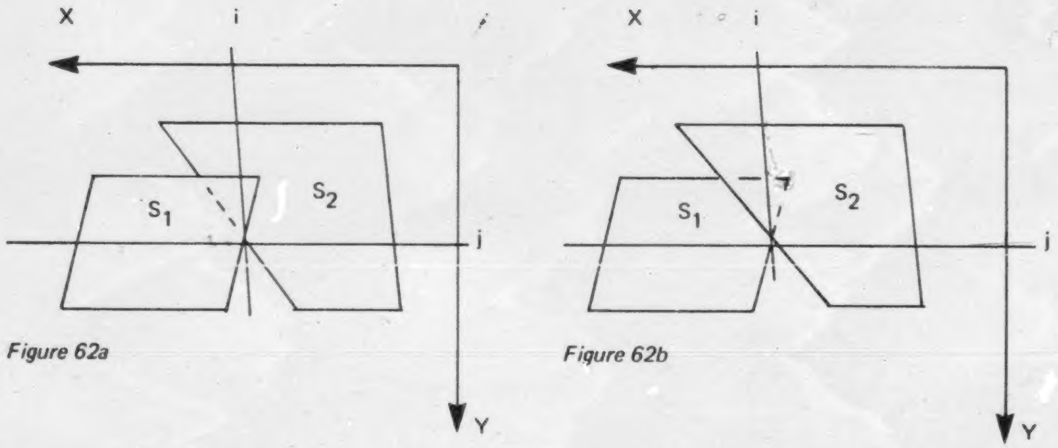


Figure 62a

Figure 62b

Figure 62 – **Choice of Edges at X = i**

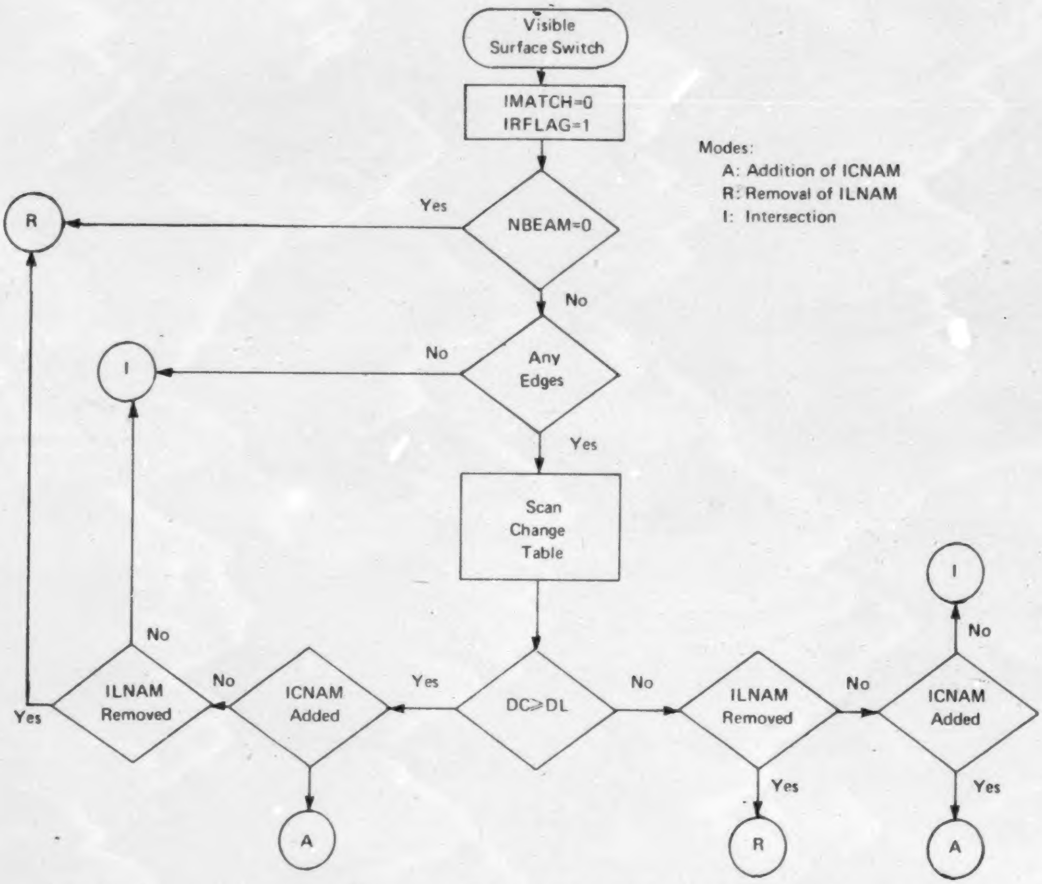


Figure 63 – **Setting Output Mode**

The intersection mode is a default mode that is entered if either there are no edges in the i^{th} X resolution interval or else none of the edges in the i^{th} resolution interval reference ICNAM or ILNAM. The partition has S_ℓ=ICNAM and S_r=ILNAM, and Slope M as calculated by [3.6]. One partition is created if ICNAM if flat-shaded; if ICNAM is smooth-shaded, a second partition is created at X=i+2. If either surface is smooth-shaded, that surface is added to the pending brightness table.

The add and the remove modes are quite similar. If both ICNAM and ILNAM are referenced by edges, DC and DL are compared to determine which of the two cases of Figure 62 has occurred. Regardless of whether the add or the remove mode has been entered, a ICNAM:ILNAM edge always has priority: If such an edge is found, IMATCH is set to point to it and its parameters are used in forming the partition. Otherwise, if the add mode is entered, the edge whose parameters are used is that edge with the smallest right-side surface label. If the remove mode is entered, the edge whose parameters are used is that edge with the smallest left-side surface label. Any error made on this scanline will be corrected on subsequent scanlines as the edges diverge into different discrete X intervals. If the edges do not diverge, the error is not significant enough to warrant correction.

Since the number of projected edges per scanlines is typically much less than R_x, the number of X resolution intervals, and since the occurrence of intersections involving front surfaces is a low-frequency event, the visible surface algorithm of the image generator has been given the ability to horizontally skip-ahead over regions void of projected edges. The initial skip-ahead attempt is to the X interval preceding the next edge. That is, if the next edge occurs at X=i+ΔX, then

$$D_k = D_k + (\Delta X - 1) D_{x_k} \quad k=1, \ldots, K \qquad [3.9]$$

where K is the number of surfaces vying for output. If ICNPNT=ILPNT, the skip-ahead was successful and the visible surface algorithm may now go back to a one unit mode after loading and unloading the beam boxes in accordance with the edges at X=i+ΔX. If ICPNT≠ILPNT, there is at least one intersection in the interval [i+ΔX,i] and the maximum likelihood estimate is that there is exactly one intersection. The point of intersection between the surface visible at X=i and the surface visible at X=i+ΔX may be calculated by equating their depths:

$$\mathrm{DSAVE}_1 + D_{x_1} \cdot \mathrm{DELTAX} = \mathrm{DSAVE}_2 + D_{x_2} \cdot \mathrm{DELTAX}$$

$$\mathrm{DELTAX} = \frac{\mathrm{DSAVE}_2 - \mathrm{DSAVE}_1}{D_{x_1} - D_{x_2}} \qquad [3.10]$$

where DSAVE is the register in each beam box remembering each depth D before skipping ahead. The next skip ahead increment is [DELTAX], the first integer less than DELTAX, and the process is repeated. Choosing [DELTAX] in this manner guarantees that the intersection will be caught between both ends of a one interval X increment. Figure 64 demonstrates how the intersection between Surface S_1, which is visible at $X=i$, and Surface S_2, which is first visible at $X=[X_K']+1$, is detected in three steps.

The first skip-ahead attempt is made at $X=j-1$, the first interval before projected Edge E4. The visible surface is S_3 necessitating the calculation of X_k, the point of intersection between surfaces S_1 and S_3. The boxes are next updated for $X=[X_k]$; here, the visible surface is still not S_1 but S_2. The point X_k' is then calculated and the beam boxes are updated for $X=[X_k']$; here, S_1 is still the visible surface and an additional one-unit increment obtains the surface ordering on the other side of the S_2:S_1 intersection.

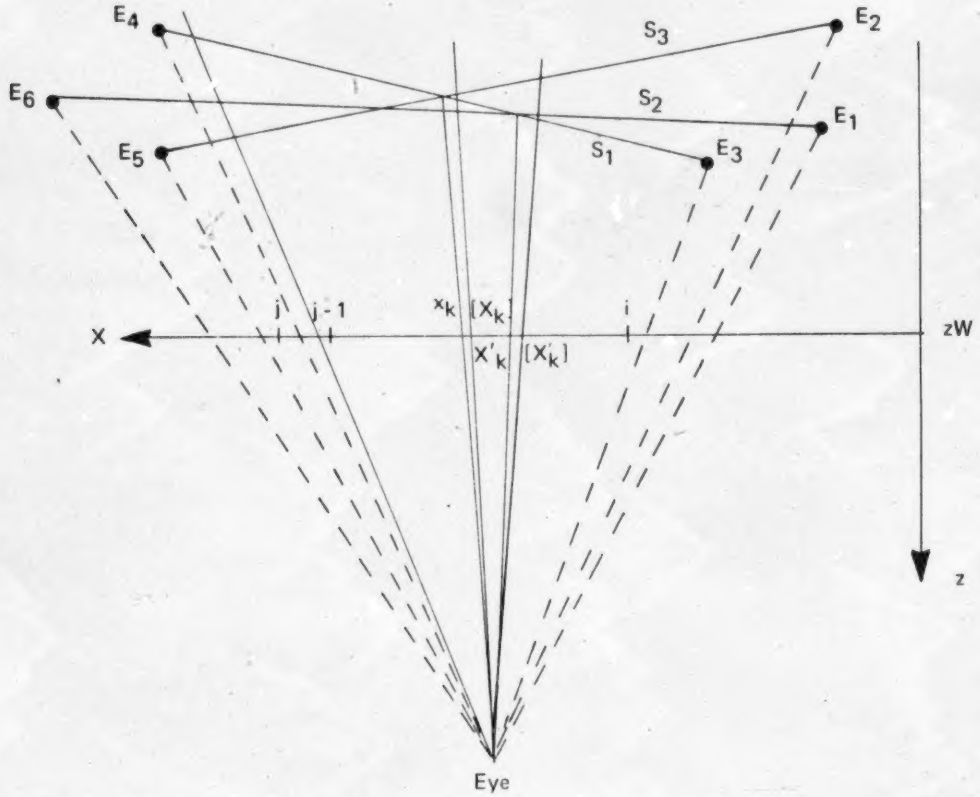


Figure 64 – **Three Step Skip-Ahead**

As each point of intersection X_k is calculated, it is checked against the maximum point $X = j - 1$ to assure that numerical inexactness (particularly with nearly coplanar surfaces) has not led to an X_k value that lies outside the permissible scanline range.

VISIBLE EDGE ENCODER

Once the X-ordered set of video partitions for scanline $Y = j$ has been formed, it is sent to the visible edge encoder to be compared against an X-ordered set of visible edges that effect similar partitions on previous scanlines. Each visible edge that is matched by a video partition and is not "too close" to the first visible edge to its right may be extended to scanline j. Each visible edge that is not matched by a video partition is not allowed to extend to scanline j but rather is terminated on a previous scanline. At the conclusion of R_y scanlines, the visible edge encoder has encoded a set of visible edges that define the entire video image.

The visible edge encoder maintains two major X-ordered tables. The first of these is the Current Visible Edge Table which holds visible edge data valid at the last processed scanline j′ and which may possibly extend to the present scanline j. The X-ordering of the Current Visible Edge Table is that of scanline j′. This X-ordering is maintained in a companion table that also maintains a listing of free locations in the Current Visible Edge Table. The size of these tables is $256 \cdot N - 1$ where N is the number of memory banks in the CHARGE terminal. Each entry in the Current Visible Edge Table is of the form of Figure 65. S_ℓ,S_r, S last TYPE, and M are as defined in the video partition originating

S_ℓ
S_r
S_{last}
TYPE
M
X
b_c
PNTR

Figure 65 – **Current Visible Edge Table Entry Format**

the visible edge. X and b_c are the horizontal position and the brightness of the edge at scanline j'. PNTR is a pointer to the location of the edge in the second table: the Current Edge Block Table.

The Current Edge Block Table holds visible edge data for all valid edges in an edge block. Some of these edges may still be current and have entries in the Current Visible Edge Table, while other edges may have ceased to be valid and are flagged as "done." Each entry in the Current Edge Block Table is of the form of Figure 66.

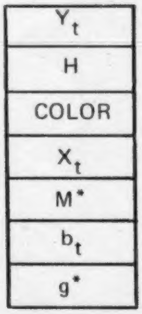


Figure 66 – **Current Edge Block Table Entry Format**

Y_t is the scanline on which the visible edge was first created; COLOR is the color to the right of the edge; X_t is the horizontal position of the edge at $Y = Y_t$; and b_t is the brightness of the edge at $Y = Y_t$. The remaining parameters are filled in when the edge ceases to be "current" and is flagged as "done." H is the height of the edge such that $Y = Y_t + H$ is the last scanline for which the visible edge is valid. M* and g* are representations of the edge's slope and brightness gradient and are given by

$$M^* = [M \cdot 2^{11-p} + .5] \tag{3.11}$$

$$g^* = [(b_c - b_t)\, 2^{11-p}/H + .5] \tag{3.12}$$

where p is the number of high order 0's in an 11-bit H. If the surface to the right of the visible edge is to be smooth-shaded (i.e., TYPE = 1 or 3), the value of Y_t is set negative flagging this condition. Companion to the Current Edge Block Table is a second table of equal length (256·N-1) which maintains the X-ordering of the visible edge block, a listing of free locations in the Current Edge Block Table, and the "done" list that flags those visible edges that have been completed and had values for H, M*, and g* calculated. The X-ordering of the Current Edge Block Table is from left to right (high X to low X) and is such that for any scanline, those edges valid on that scanline are X-ordered between themselves. The X-order between groups of edges disjoint in Y-ranges is arbitrary.

After a block of visible edges is complete, the visible edge encoder sends it to an output table in sequential left-to-right order preceded by a marker edge indicating the Y-range over which the block of edges is valid (see Figures 46-48). The size of the output table is the size of the CHARGE terminal's memory.

The last table that the visible edge encoder uses is a sequential right-to-left X-ordered Scanline Output Table remembering the X-order of current visible edges as they are encountered on Scanline j. Each item in this table has three entries:

(1) The location of the edge in the Current Visible Edge Table

(2) A bit to say if the edge is "new" (i.e., if the edge has been created on Scanline j)

(3) A pointer to the edge in the Current Edge Block Table that is the first edge to its right on scanline j. This table is used for updating the X-ordered pointers for the Current Visible Edge Table and the Current Edge Block Table after the matching processes have been completed.

Figure 67 provides a functional flowchart of the operations on scanline $Y = j$. If $j = 1$, the display is initialized clearing out all of the visible edge tables and setting $Y_{last} = 0$ (where Y_{last} is the last scanline processed by the visible edge encoder) and $Y_{block} = 1$ (where Y_{block} is the initial valid scanline for which the current edge block is valid). Next, $\Delta Y = j - Y_{last}$ allowing positional updates of current visible edges ($X_i = X_i + \Delta Y \cdot M_i$). A test as to whether or not the current edge block should be ended and another begun is then made. This test is

$$N \cdot (256 - [(Y_{block} - j)/40] - 1) > N_{CB} + N_V \qquad [3.13]$$

where the left side of the inequality represents the maximum number of non-marker edges in a visible edge block (see [3.3]), and the right side represents the maximum number of potential visible edges in the current edge block at the end of processing for scanline $Y = j$. Here, N_{CV} is the number of visible edges presently in the Current Edge Block Table (both current edges and "done" edges) and N_V is the number of video partitions. Obviously, none of the visible edges may match any of the video partitions yielding $N_{CB} + N_V$ as the maximum possible number of visible edges at the end of processing for scanline $Y = j$. Initial scanline processing is completed with a check for end of display (i.e., $Y > R_y$). If the end of display has been reached, the current edge block is completed and sent to the output table. Then the rest of the output

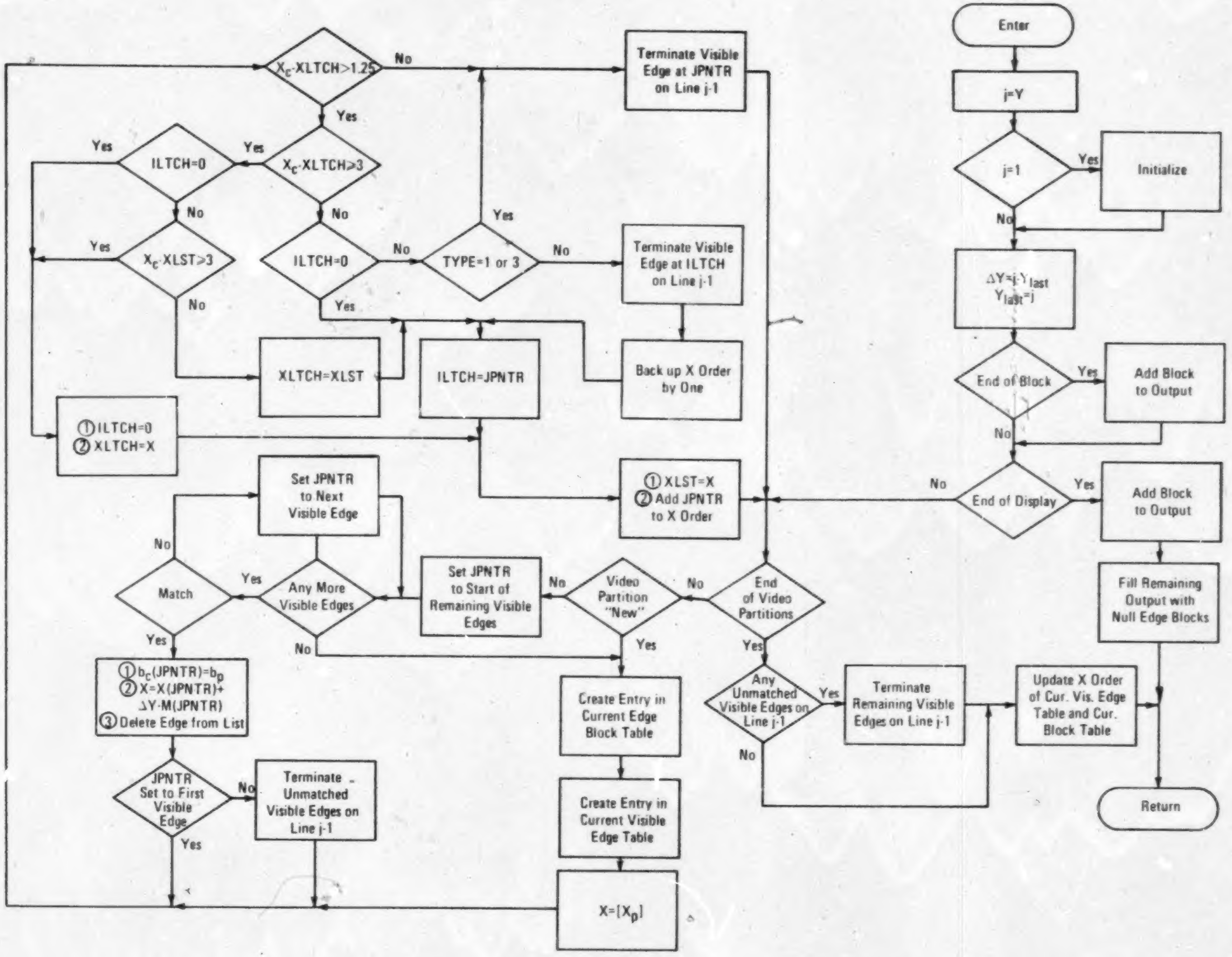

Enter
j=Y
j=1
Yes
Initialize
No
ΔY=j-Y last
Y last=j
End of Block
Yes
Add Block to Output
No
End of Display
Yes
Add Block to Output
No
Fill Remaining Output with Null Edge Blocks
Update X Order of Cur. Vis. Edge Table and Cur. Block Table
Return
X_c-XLTCH>1.25
No
Yes
Terminate Visible Edge at JPNTR on Line j-1
X_c-XLTCH>3
Yes
ILTCH=0
Yes
No
X_c-XLST>3
Yes
No
No
ILTCH=0
No
Yes
TYPE=1 or 3
Yes
No
Terminate Visible Edge at ILTCH on Line j-1
Back up X Order by One
XLTCH=XLST
ILTCH=JPNTR
① ILTCH=0
② XLTCH=X
① XLST=X
② Add JPNTR to X Order
End of Video Partitions
No
Yes
Any Unmatched Visible Edges on Line j-1
Yes
No
Terminate Remaining Visible Edges on Line j-1
Video Partition "New"
No
Yes
Set JPNTR to Start of Remaining Visible Edges
Any More Visible Edges
Yes
No
Set JPNTR to Next Visible Edge
Match
No
Yes
① b_c(JPNTR)=b_p
② X=X(JPNTR)+ ΔY·M(JPNTR)
③ Delete Edge from List
JPNTR Set to First Visible Edge
No
Yes
Terminate Unmatched Visible Edges on Line j-1
Create Entry in Current Edge Block Table
Create Entry in Current Visible Edge Table
X=[X_p]

Figure 67.— **Visible Edge Encoder**

table is filled with null edge blocks that serve both to (a) fill up the remainder of the CHARGE terminal's memory and (b) provide null data for those Y values (greater than R_y) that represent vertical retrace time between interlaces.

Once initial scanline processing has been completed, the major processing loop of matching the video partitions and the visible edges is entered. The procedure is to attempt to match each video partition (working in X order from right to left) with an unmatched visible edge (scanning from right to left). The parameters used in matching are S_ℓ, S_r, S_{last}, and TYPE. If no match is possible, a new visible edge is created. If the visible edge that is matched is not the first of the previously unmatched visible edges (X-ordered from right to left as they appeared on scanline j′, the last scanline worked on by the visible edge encoder), all of the visible edges up to the matched visible edge are terminated on scanline j- 1 since any subsequent attempt to extend one or more of them would lead to an illegal "crossover" condition (i.e., there would not exist a valid X-ordering of the edges in the Current Edge Block Table). The brightness and the horizontal position of the visible edge are updated to scanline j by setting $b_c = b_p$ and $X = X + \Delta Y \cdot M$. Of course, if the visible edge has just been created on Scanline j, then $X = [X_p]$.

The CHARGE terminal uses M*, a less precise value of M, as it decodes the visible edges into an image, In Appendix C it is shown that for $R_x < 2048$ and $R_y < 1024$, a 13-bit M* (12 bits plus sign bit) can lead to a maximum deviation of $e = 0.5$ units in X. Note, too, in Figure 68 that there is an error or up to 1 unit in X in the position of X_t between the original projected edge and the visible edge. The error is due to the necessity of starting the edge at the CHARGE terminal (and, hence, the visible edge encoder) at the discrete $X = [X_p]$ rather than at $X = X_p$. The design of the CHARGE terminal is such that decoded edges may never cross each other; they may come as close as 1 unit in X, but they should maintain an average separation of two units in X between every three edges.

Accordingly, after an edge has been matched with a video partition and tentatively extended to scanline j, it must be compared in closeness to the last visible edge to its right to make certain that there is no danger of a crossover at the CHARGE terminal. Hence, after an edge has been matched, it enters the processing of Figure 67. Two registers are used to remember X information about previous edges on the scanline. XLST contains the X position of the last visible edge to the right; XLTCH contains the X position of the last visible edge to the right that satisfied the requirement of an

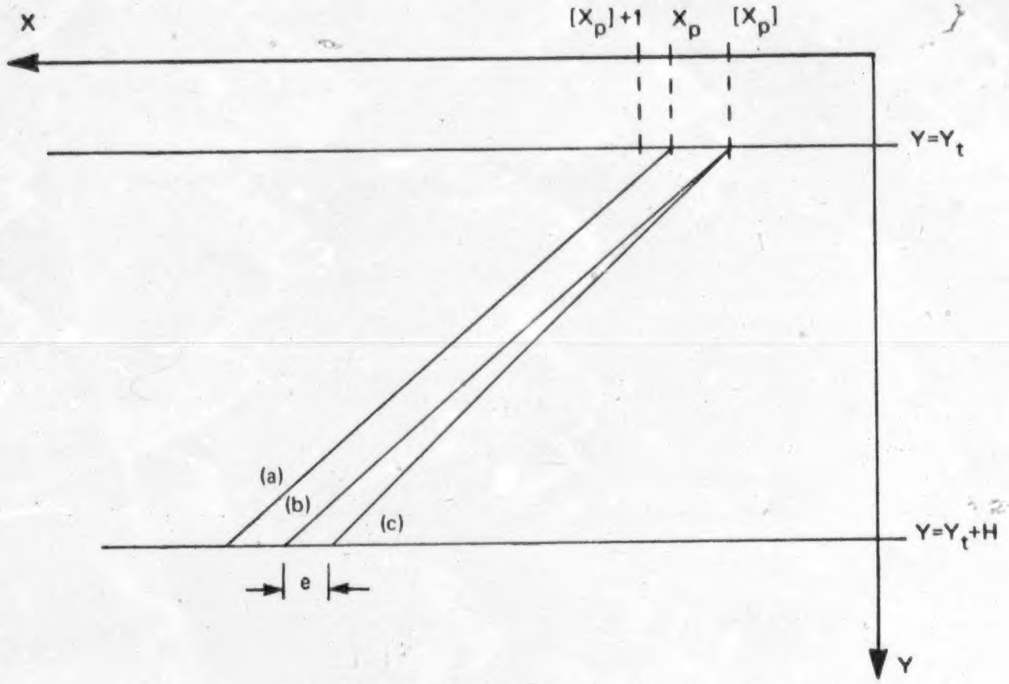


Figure 68 – **Projected Edge; Edge at Visible Edge Encoder; and Edge at CHARGE Terminal**

average separation of two units in X for the last three edges. If (XLST - XLTCH) < 3 where XLST ≠ XLTCH, ILTCH points to the edge at XLST which has been tentatively extended to scanline j but which may be deleted if the present edge, positioned at $X = X_c$, is "too close."

Figure 69 demonstrates the normal mode of operation on scanline Y = j. The last line worked on was $Y = Y_{last}$. Seven scanline partitions on line Y_{last} generated seven visible edges in the Current Visible Edge Table. These edges have the order: a, b, c, d, e, f, g. If no other visible edges are present in the Current Block Table, the entries in Current Block Table have the order: g, f, e, d, c, b, a. Note that a visible edge/scanline partition always is present at X = 1 and $X = R_X$, due to the 1:0 and 0:1 projected edges present in every image.

As the processing of scanline Y = j begins, the first partition immediately matches current visible edge "a." The next two video scanline partitions immediately match current visible edges "b" and "c." As each visible edge is matched, it is removed from the X-ordered list of current visible edges yet to be matched. The fourth video scanline partition is then compared with the next visible edge, "d," and it is found that they do not match; however,

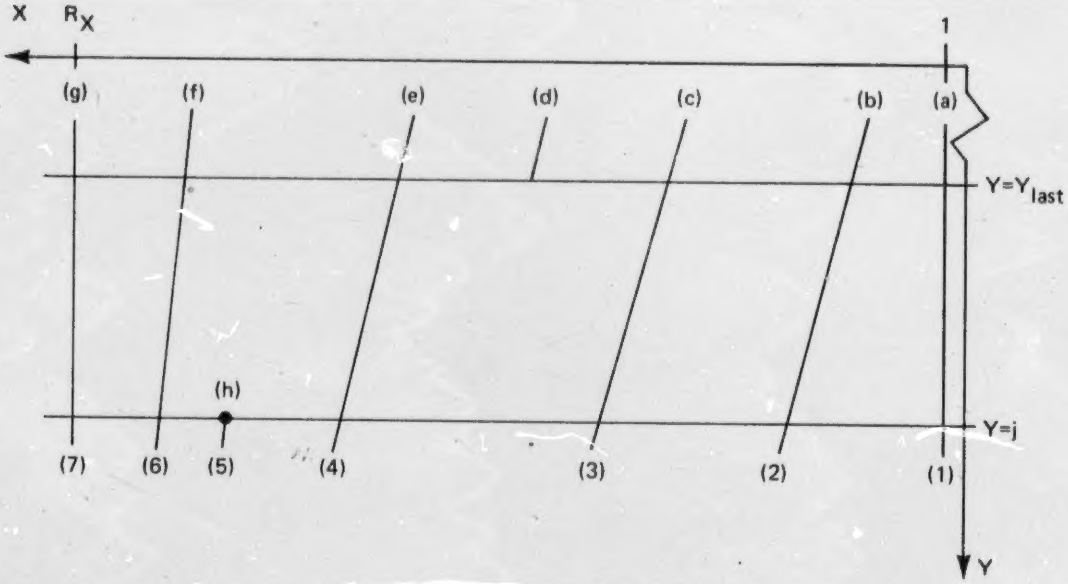


Figure 69 – **Visible Edge Encoding on Scanline Y=j**

it is found that partition “4” matches visible edge “e.” Since edge “e” is not the first visible edge in the X-ordered current visible edge list, it is deemed that those edges with lesser X-values than edge “e” (here, only edge “d”) are no longer valid and should be terminated on scanline Y_{last}.

Next, partition “5” is encoded. If partition “5” is tagged as “new” (i.e., it is the top of a projected edge), a new visible edge is immediately created. If, however, it is not tagged as “new,” then partition “5” is compared with all of the remaining current visible edges in search of a match before a new visible edge is created.

Finally, partitions “6” and “7” match current visible edges “f” and “g.” The Current Visible Edge Table now has the X-order: a, b, c, e, h, f, g. The Current Edge Block Table has the X-order: g, f, h, e, d, c, b, a. Figure 70 demonstrates the X-ordering of a block containing edges for a cube.

VERTICAL SKIP-AHEAD

There is, in general, a great deal of coherence between successive scanlines on which no new projected edges become valid or no projected edges have ceased to be valid. This is due to the fact that a surface may as well be thought of as generating vertical Y-ordered partitions over which it is valid for a given horizontal X value, rather than horizontal X-ordered partitions over which it is valid for a given vertical Y value. Simply stated,

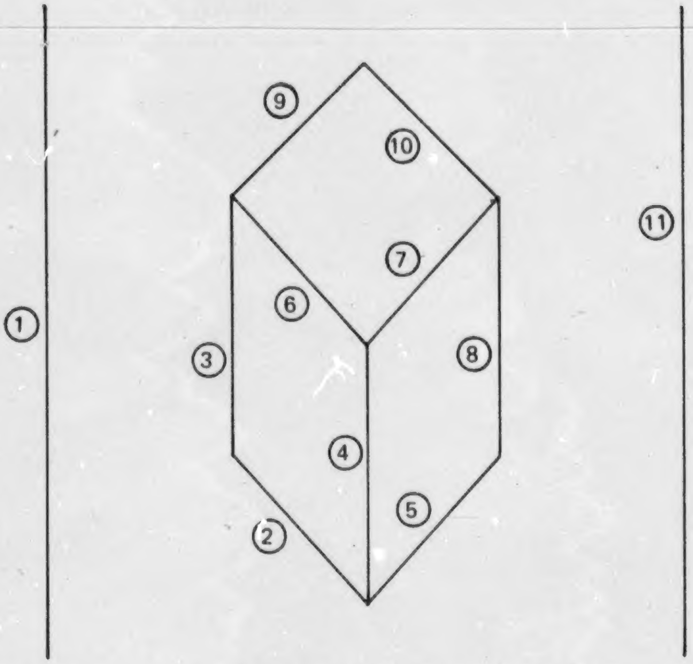


Figure 70 – **Order of Edges for Block Depicting a Cube.**

visible surfaces tend to remain visible unless the edges defining them end or unless new edges defining new visible surfaces begin.

Each scanline processing step requires the following operations:

(1) Adding new projected edges to the set of current projected edges.
(2) Updating the current projected edge set for the next scanline. This processing includes deletion of edges that cease to be valid (i.e., H goes negative upon update).
(3) Generating actual video scanline partitions.
(4) Possible readout of a completed visible edge block.
(5) Updating the current visible edge set.

Furthermore, it is known that at the completion of a scanline processing step, the actual X-ordered video partitions are matched exactly by the current set of visible edges if no deletions of visible edges occurred due to the "closeness" comparison test. Hence, if the order of the actual video partitioning of the scanline remains constant over a skip-ahead attempt of Δ Y scanlines over which the set of valid projected edges is constant, then the skip-ahead attempt is known to be successful.

If no edges start or end on a given scanline, the above set of operations reduces to:

(1) Updating the current projected edge set by updating D_0 and X and maintenance of the X-ordered edge list.

(2) Generating the actual video scanline partition.

(3) Updating the current visible edge set.

Vertical skip-ahead attempts require the following processing:

(1) Saving the current projected edge set and its X-ordered list.

(2) Saving the actual video scanline position.

(3) Updating the current visible edge set by updating D_0 and X and generating an X-ordered edge list.

(4) Generating the actual video scanline partitions.

(5) Comparing this new scanline partition with the saved scanline partition.

(6) If the skip-ahead attempt is successful, the parameters of the projected edge set and the X-ordering of the projected edge set must be replaced by those that are valid at the skip-ahead Y value.

If multiple skip-ahead attempts are made within a range of scanlines, operations (3), (4), and (5) are repeated with each attempt.

Generally speaking, the average amount of time required to process Δ Y scanlines without any vertical skip-ahead is given by

$$T_0 = \Delta Y \cdot E[t_0(n)] \qquad [3.14]$$

where

n : Number of edges in the current projected edge set.

$t_0(n)$: Random process mapping n into an estimate of processing time required to process n projected edges into actual video scanline partitions and then into X-ordered visible edges.

$E[\cdot]$: Expectation operator.

If a vertical skip-ahead is successful over the same Δ Y scanlines, the average amount of time required is given by

$$T_1 = T_s(n) + E[t_1(n)] \qquad [3.15]$$

where $T_s(n)$ is a setup time and $t_1(n)$ is a random processing time ($E[t_1(n)] > E[t_0(n)]$). Let two hypotheses be:

H_0 : Vertical skip-ahead will be successful with probability P.

H_1 : Vertical skip-ahead will not be successful with probability (1-P).

The cost matrix for the vertical skip-ahead decision problem is of the form

$$C = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} \qquad [3.16]$$

where c_{ij} is the cost of choosing H_i when H_j is actually true. Here,

$$C = \begin{bmatrix} t_0 & t_0 \\ T_0 + T_1 & T_1 \end{bmatrix} \qquad [3.17]$$

The average risk is given by

$$\begin{aligned} \overline{C} = {} & P[c_{00}P_r \text{ (choose } H_0, H_0 \text{ true)} + c_{10}P_r \text{ (choose } H_1, H_0 \text{ true)}] \qquad [3.18] \\ & + (1\text{-}P)\,[c_{01}P_r \text{ (choose } H_0, H_1 \text{ true)} + c_{11}P_r \text{ (choose } H_1, H_1 \text{ true)}] \end{aligned}$$

The problem of minimizing [3.18] is quite complex. Two approaches have been considered.

APPROACH I: If the joint probability density functions (p.d.f.) $p_0\,(n, \Delta y)$ and $p_1\,(n, \Delta y)$ for the two hypotheses H_0 and H_1 were known a priori, it would be possible to form the likelihood ratio

$$\Lambda\,(n, \Delta y) = \frac{p_1\,(n, \Delta y)}{p_0\,(n, \Delta y)} \qquad [3.19]$$

which would be compared to the threshold value

$$\Lambda_0\,(n, \Delta y) = \frac{P\,(c_{10} - c_{00})}{(1-P)\,(c_{01} - c_{11})} = \frac{P}{(1-P)}\,\frac{T_1}{T_0 - T_1} \qquad [3.20]$$

The strategy would then use the Bayes decision rule

$$\begin{aligned} & \Lambda < \Lambda_0 \text{ , chose } H_0 \\ & \Lambda > \Lambda_0 \text{ , chose } H_1 \qquad [3.21] \\ & \Lambda = \Lambda_0 \text{ , chose either } H_0 \text{ or } H_1 \end{aligned}$$

Hence, to implement APPROACH I it is necessary to analyze a vast number of images, gathering data on values of (n, Δ y) for which vertical skip-ahead was and was not successful. From these data, an empirical decision surface could be defined. (see Figure 71).

APPROACH I has not been implemented because it was feared that any particular sample of test images would not be sufficiently varied as to converge upon the true Bayes risk; that is, any particular set of images would unintentionally weight the decision surface in one way or another.

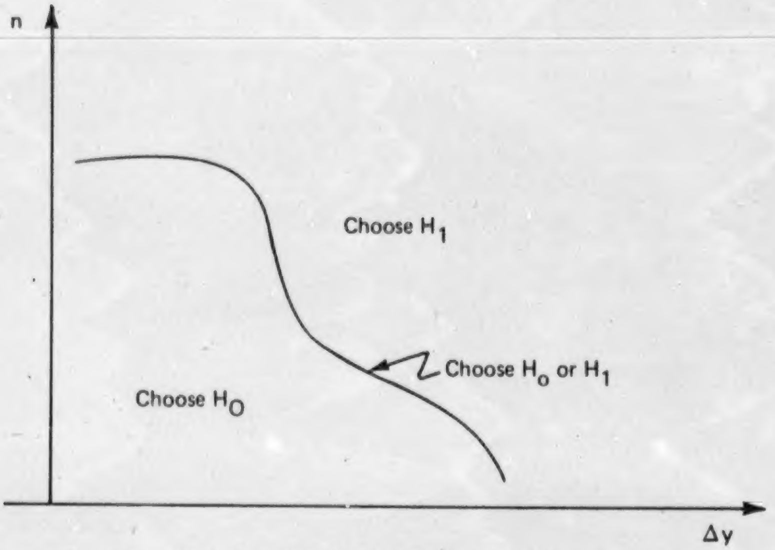


Figure 71 – **Empirical Skip-Ahead Decision Rule.**

APPROACH II: The second skip-ahead approach seeks to make use of certain known properties of projected atoms and objects via a sophisticated, but suboptimum, skip-ahead strategy. These known properties are:

(1) If $n = 2$, only the two border edges at $X = 1$ and $X = R_X$ are present. Hence, if $n = Z$, skip-ahead will always succeed.

(2) For n large, $T_1 \gg T_0/\Delta Y$; that is, the time required in a skip-ahead attempt becomes much larger than the average time per scanline for normal processing. Also, if n becomes large, the probability of placing n projected edges on a scanline without changing the order of visible edges becomes small. Hence, if $n > \theta_n$, a threshold, vertical skip-ahead attempts are prohibited.

(3) Since $T_1 > T_0$, for $\Delta Y < \theta_y$, a threshold, it does not really pay to use vertical skip-ahead.

(4) Actual video scanline partitions change from scanline to scanline because either (a) the order of visible surfaces changes or (b) projected edges come "too close." The former case is roughly related to the number of distinct atoms in the region Δ y (and hence is related linearly to n); such crossover cases occur with equal probability throughout Δ y. The latter case occurs at the vertices of atom facets (i.e., scanlines on which projected atom edges are started or are terminated). Thus, a reasonable probability density function for a successful vertical skip-ahead might look like Figure 72A, while possible p.d.f.'s for vertical skip-aheads that fail are like Figure 72B-E. That is, if a

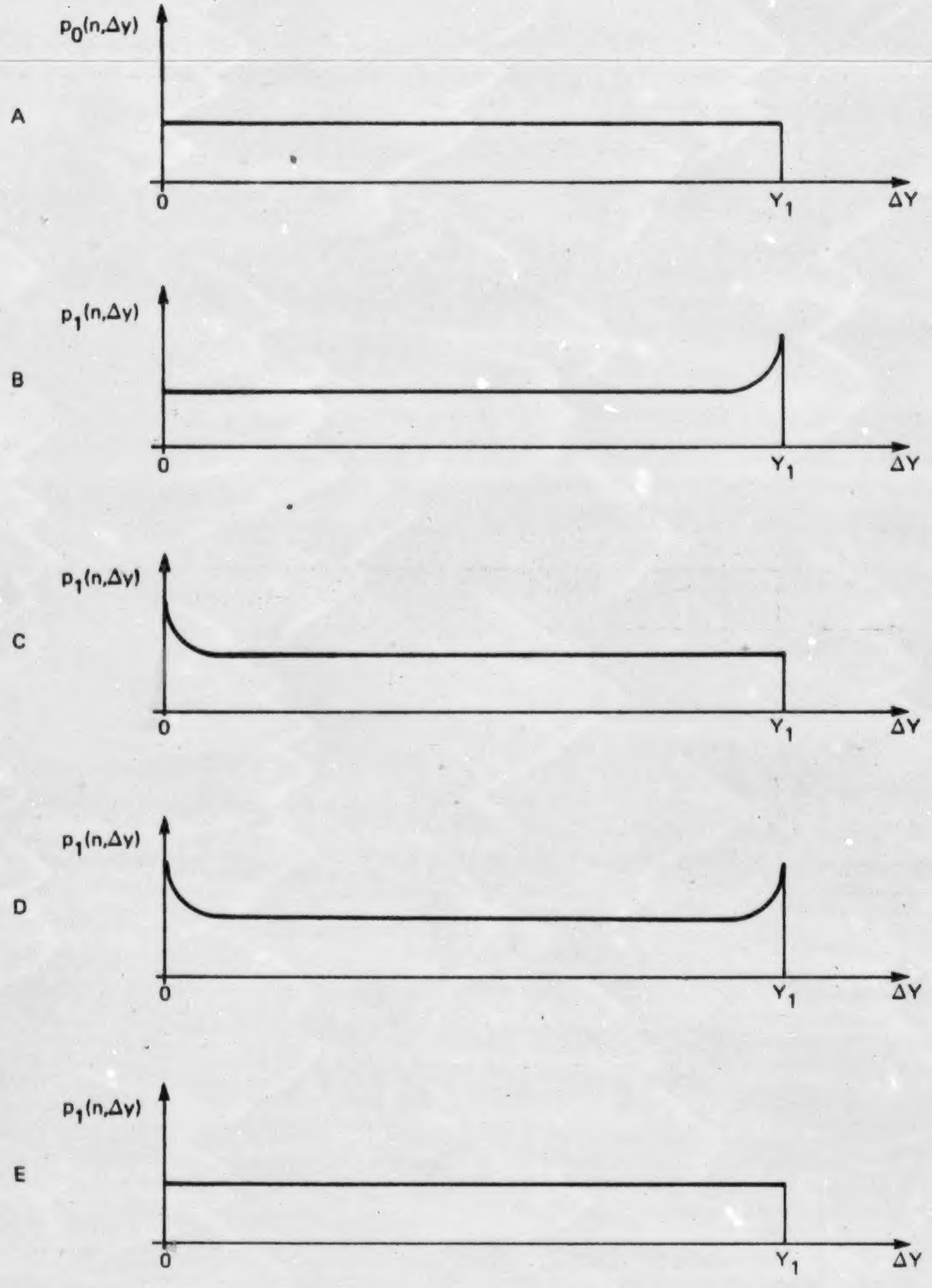


Figure 72. **Probability Density Functions for Vertical Skip-Ahead.**

skip-ahead attempt of $\Delta Y = Y_1$ scanlines should fail, the most likely place for it to fail (i.e., $p_1(n, \Delta y)/p_0(n, \Delta y) = \max$) is at one or both end points in Y_1.

Hence, APPROACH II uses the following algorithm:

(i) If $n = 2$, always perform a skip-ahead.

(ii) If $n > \theta_n$ never attempt a skip-ahead.

(iii) If $\Delta Y < \theta_y$, do not attempt a skip-ahead.

(iv) Otherwise, attempt a skip-ahead of the full $\Delta y = Y_1$ scanlines.

(v) If an attempted skip-ahead should fail, proceed according to the functional flow of Figure 73. This algorithm, having first failed to accomplish the maximum skip-ahead (Figure 72A), successfivly attempts to decide whether p_1 $(n, \Delta y)$ is of the type shown in Figure 72B, 72C, 72D, or 72F. That is, having failed at $\Delta Y = Y_1$, it next attempts at $\Delta Y = Y_1/2$ should fail, the p.d.f. is probably either type C, D, or E and the next attempt is at $\Delta Y = \theta_y$. The process continues each step remembering what has been tried before until the process is complete. At each stage of the process, rule (iii) applies.

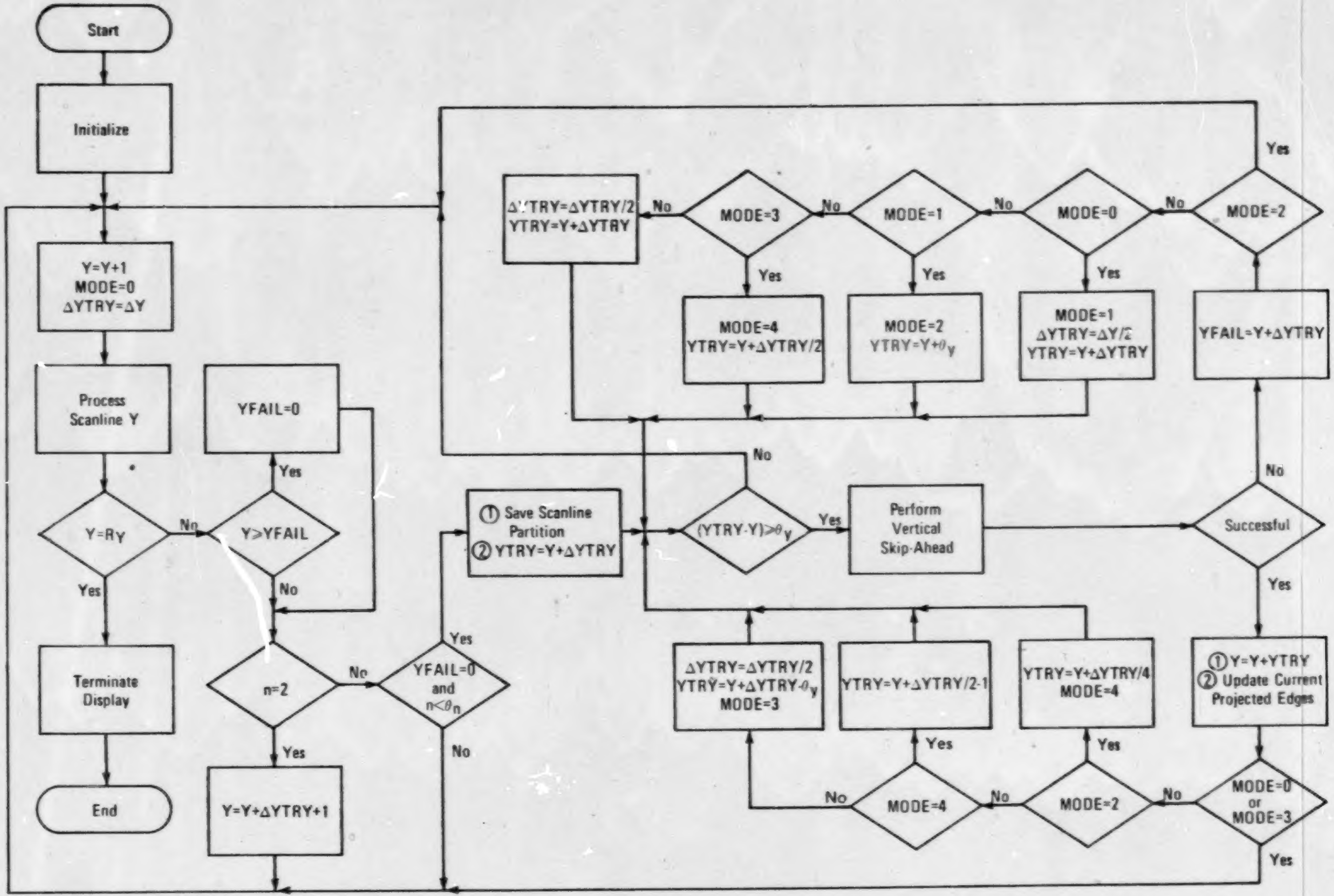


Figure 73— **Vertical Scanline Processing**

APPENDICES

Appendix A

EXAMPLE OF A NON-PLANAR SURFACE

Consider the right cylinder of radius R and height H in Figure A-1.

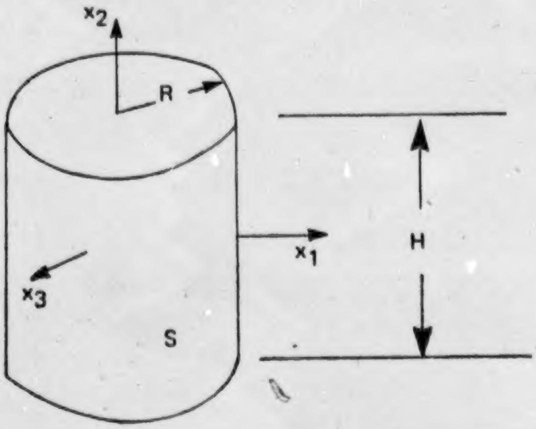


Figure A-1 – Right Cylinder

The points on the curved Surface S, satisfy

$$f(\bar{x}) = 0 = x_1^2 + x_3^2 - R^2, \qquad -\frac{H}{2} < x_2 < \frac{H}{2} \qquad \text{[A-1]}$$

The gradient vector at any of these points on S is given by

$$\bar{N}(\bar{x}) = \Delta f^T(\bar{x}) = (2x_1, 0, 2x_3)^T \qquad \text{[A-2]}$$

The problem is to represent the closed surface S with a set of $n>2$ planar surfaces. The approximation error at any point $\bar{x}$ on the original surface S will be defined as the distance between $\bar{x}$ and $\hat{x}$, the closest point to $\bar{x}$ on the approximating surfaces $\left\{\hat{S}_i\right\}$. That is

$$e(\bar{x}) = \min\left\{ \|\bar{x} - \hat{x}_i\| \;\middle|\; \hat{x}_i \in \hat{S}_i \right\} \quad i=1,\ldots,n \qquad \text{[A-3]}$$

Since the gradient vector $\bar{N}(\bar{x})$ for S happens to be independent of x_2, any surface $\hat{S}_i$ representing a portion of S should also have a gradient vector $\hat{N}_i(S_i)$ which has a zero x_2 component to minimize $e(\bar{x})$ over $\hat{S}_i$; that is, any surface $\hat{S}_i$ may be extended in the

x_2 direction without incremental error (i.e., without adding to the norm $\|\bar{x} - \hat{x}_i\|$). This means that the most efficient set of n planar surfaces is a set of n rectangles as in Figure A-2. The rotational separation between adjacent facets is $\theta = 2\pi/n = \pi/4$. Also note that rather than just the end points, each of the vertical edges in Figure A-2 lies entirely in S. This is an additional benefit of having $\overline{N}(\bar{x})$ independent of x_2 (i.e., curvation in only 2 of the 3 dimensions).

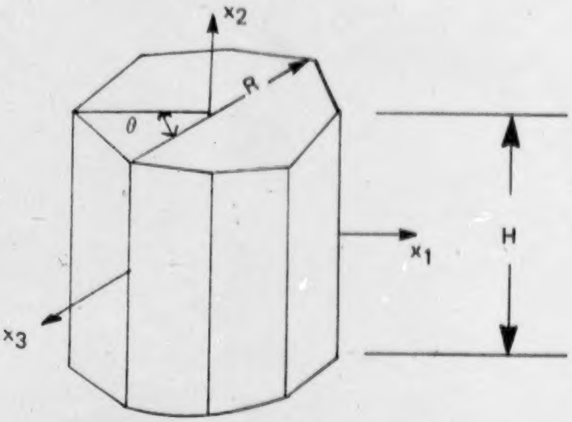


Figure A-2 – **Right Cylinder Represented by n = 8 Planar Surfaces**

For the right cylinders of Figures A-1 and A-2, the error for any one facet $\hat{S}_i$ (at x_2 = constant) is as shown in Figure A-3. $\bar{x}$ is a point on S; $\hat{x}$ is the closest point on $\hat{S}$; and $\bar{x}_1$ and $\bar{x}_2$ are points on the vertical edges of $\hat{S}$. $\hat{S}$ has been arbitrarily located such that it is bisected by the $x_3 = 0$ plane. The error is given by

$$e(\phi) = \|\bar{x} - \hat{x}\|$$

$$= \| R\cos\phi - R\cos\theta/2 \| \qquad -\theta/2 \leq \phi \leq \theta/2$$

$$e(\phi) = R\,|\cos\phi - \cos\theta/2| \qquad -\theta/2 \leq \phi \leq \theta/2$$

In terms of n rectangular facets

$$e(\phi) = R\,(\cos\phi - \cos\pi/n) \qquad -\pi/n \leq \phi \leq \pi/n \qquad [A\text{-}4]$$

and the maximum error over any one facet is given by

$$e_{max} = e(0) = R\,(1 - \cos\pi/n) \qquad [A\text{-}5]$$

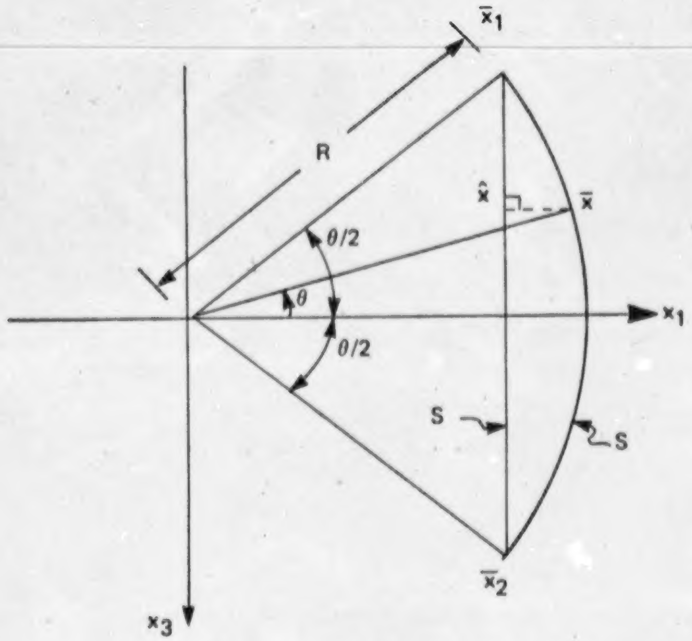


Figure A-3 – $e(\phi) = e(\bar{x}) = \|\bar{x} - \hat{x}\|$

The normalized error of the approximation of S by $\left\{\hat{S}_i\right\}$ is given by $e_N = e/R$.

$$e_N(\phi) = \cos\phi - \cos\pi/n \qquad -\pi/n \leq \phi \leq \pi/n \tag{A-6}$$

$$e_{N_{max}} = 1 - \cos\pi/n \tag{A-7}$$

Figure A-4 demonstrates the reduction of $e_{N_{max}}$ as the number of approximating surfaces is increased. It should be noted that after an initial rapid improvement in the approximation, the marginal improvement in the approximation becomes quite small.

The brightness at any point $\bar{x}$ on Surface S due to a sunlight source at infinity (i.e., parallel light rays of constant intensity through R^3) is given by

$$B(\bar{x}) = B_s f\left(\frac{\bar{N}^T(\bar{x}) \cdot \bar{x}_s}{\|\bar{N}(\bar{x})\| \; \|\bar{x}_s\|}\right) \tag{A-8}$$

where B_s is a scalar measure of the sunlight intensity, $\bar{x}_s$ is the direction of the sunlight through R^3, $\overline{N}(\bar{x})$ is the normal vector to surface S at point $\bar{x}$, and $f(\alpha)$ is the brightness

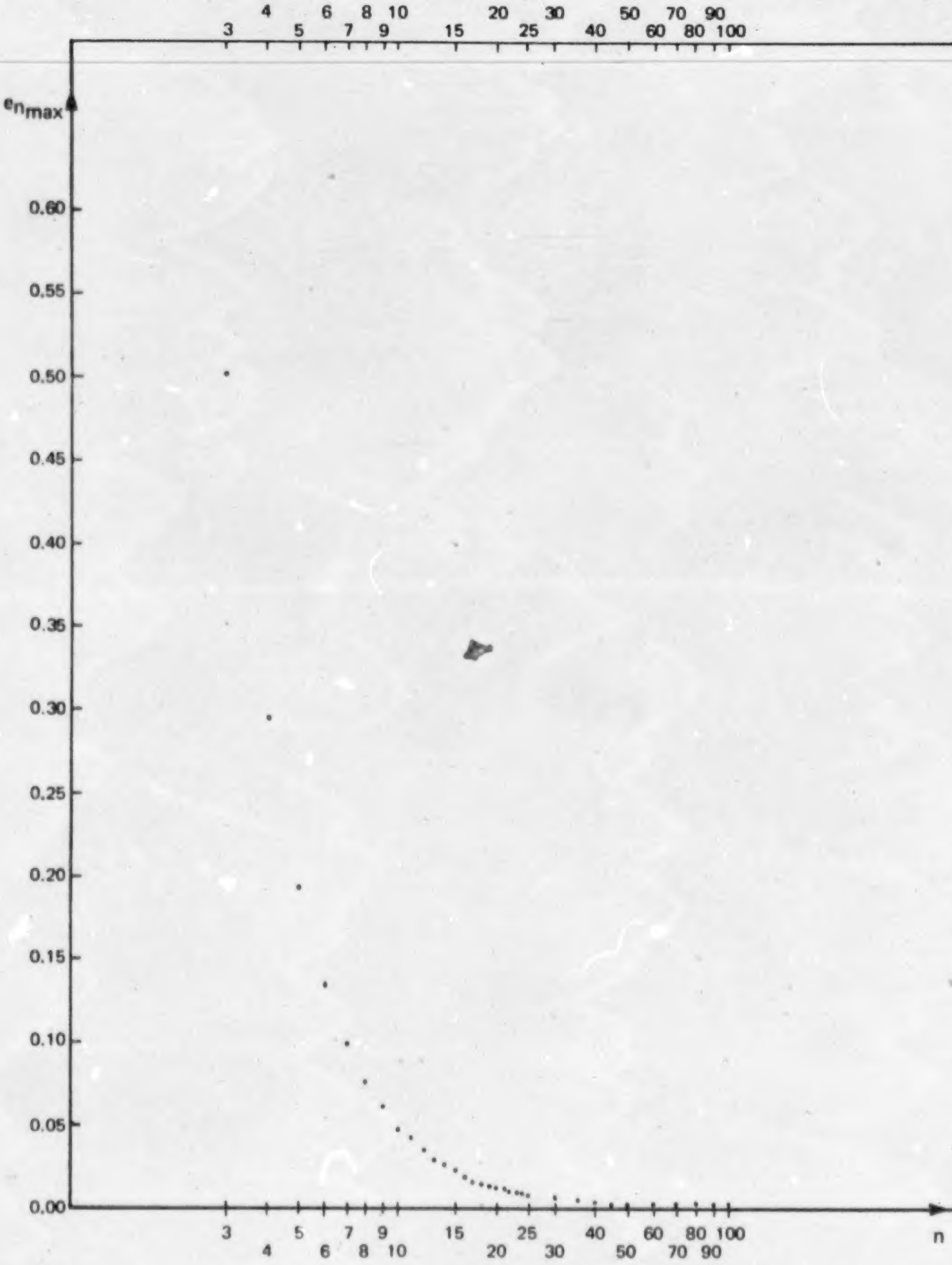


Figure A-4 — **Reduction of $e_{n_{max}}$ by Increasing the Number of Cylinder Facets** reflectivity function for surface S. A typical brightness reflectivity function for a nonshiny surface is given by

$$f(\alpha) = \begin{cases} \dfrac{A_1 + A_2 + \cos\alpha}{A_1 + A_2 + 1} & \text{if } |\alpha| \leq \pi/2 \\ \dfrac{A_1 + A_2 + A_2\cos\alpha}{A_1 + A_2 + 1} & \text{if } |\alpha| > \dfrac{\pi}{2} \end{cases} \qquad \text{[A-9]}$$

where A_1 and A_2 represent (as fractions of the direct light reflectivity) the effects of diffuse light in R^3. A_1 is an overall ambient light that affects the entire object; that is, it adds light to all surfaces equally whether or not they face toward the sun. A_2 represents the effects of reflected light from surrounding objects and provides the smooth-shading of the dark-side object surfaces (i.e., those surfaces that face away from the sun). Hence, the reflectivity function, $f(\alpha)$, falls off as the cosine of α from a peak value of 1 at $\alpha = 0$ to a value proportional to $A_1 + A_2$ at $\alpha = \pi/2$. As α increases from $\pi/2$ to π, $f(\alpha)$ falls off as the cosine of α to a baseline value proportional to A_1.

If $A_1 = 0$, those surfaces pointing directly away from the sun are black. If $A_2 = 0$, all surfaces facing away from the sun have equal brightness. If $A_1 = A_2 = 0$, the effect to the eye is as if the object were in outer space with bright sunlit surfaces, and black surfaces which face away from the sun. Typical values for A_1 and A_2 on a clear, bright day are $A_1 = .05$ and $A_2 = .3$. Figure A-5 demonstrates the behavior of [A-9].

Again consider the curved surface S of the right cylinder of Figure A-1 and the set of approximating surfaces $\{\hat{S}_i\}$ of the approximate right cylinder of Figure A-2. Due to symmetry about the x_2 axis, the brightness shading of both S and $\{\hat{S}_i\}$ is independent of the x_2 component of $\overline{x}_3$. Hence, the simpler, 2-dimensional case of Figure A-6 may be considered. $\overline{x}'_3$ and $\overline{N}'(\overline{x})$ are the (normalized) projections of $\overline{N}(\overline{x})$ and $\overline{x}_s$ onto a constant x_2 plane. The angle ϕ between $\overline{x}'_3$ and $\overline{N}'(\overline{x})$ is given by $\cos^{-1}(\overline{N}'^T(\overline{x}) \cdot \overline{x}'_3)$. The brightness at any point $\overline{x}$ on S is given by [A-8] and is merely a scaled plot of $f(\phi)$. However, the precise brightness for $\{\hat{S}_i\}$ is known only at the n points (equally spaced, but randomly phased W.R.T. ϕ) in $\phi \in [0, 2\pi]$. The brightness between pairs of these n points is given by either the linear rule

$$B(\lambda) = B(i) + \lambda\,(B(i+1) - B(i)) \qquad \lambda \in [0,1] \qquad \text{[A-10]}$$

or the logarithmic rule

$$B(\lambda) = B(i)\left(\frac{B(i+1)}{B(i)}\right)^{\lambda} \qquad \lambda \in [0,1] \qquad \text{[A-11]}$$

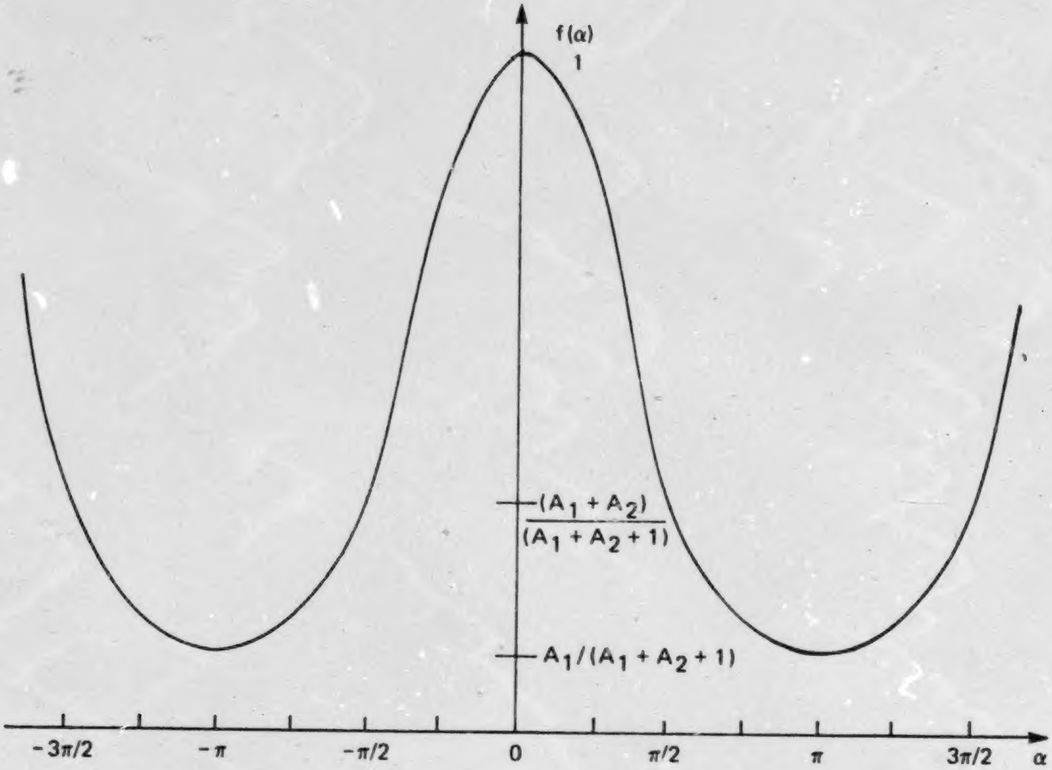


Figure A-5 – **Typical Brightness Reflectivity Function**

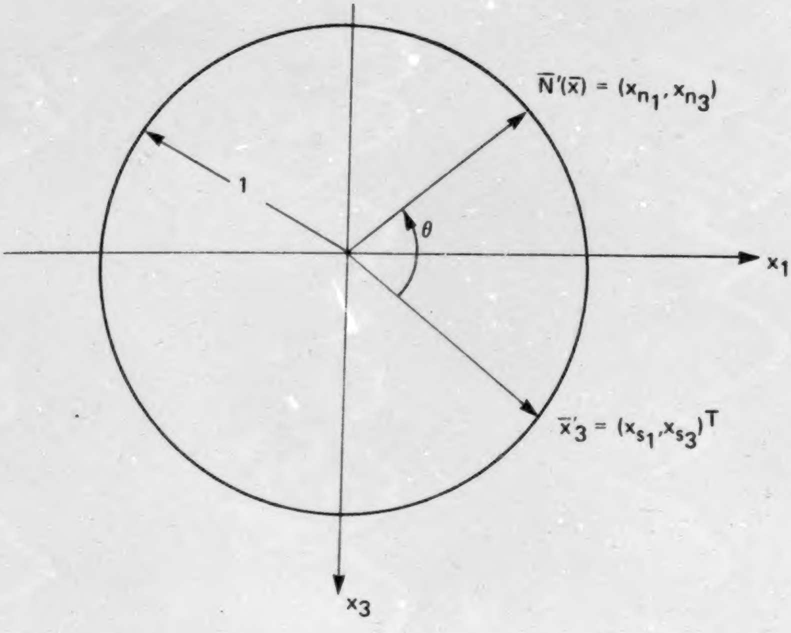


Figure A-6 – **Smooth Shading Equivalent**

The actual brightness, $B(\phi)$, and two approximating linear brightness functions, $\hat{B}_1(\phi)$ and $\hat{B}_2(\phi)$, phased the maximum distance of $\theta = \pi/n = \pi/8$ radians apart, are plotted in Figure A-7.

It should be noted that both $\hat{B}_1(\phi)$ and $\hat{B}_2(\phi)$ use the same brightness approximation rule in shading $\left\{\hat{S}_i\right\}$ over n=8 points in the angular range $[0, 2\pi]$. It is simply the chance orientation of the 8 facets W.R.T. the sun vector that causes $\hat{B}_1(\phi)$ to be a closer approximation than $\hat{B}_2(\phi)$. Not only does $\hat{B}_2(\phi)$ fail to reflect the true brightness extremes of $B(\phi)$, but it also erroneously indicates regions of constant brightness at $\phi = \pm k\pi$, $k = 0, 1, 2, \ldots$. Hence, $\hat{B}_2(\phi)$ is the worst case linear estimate for $B(\phi)$ and therefore should be used in any brightness error evaluations.

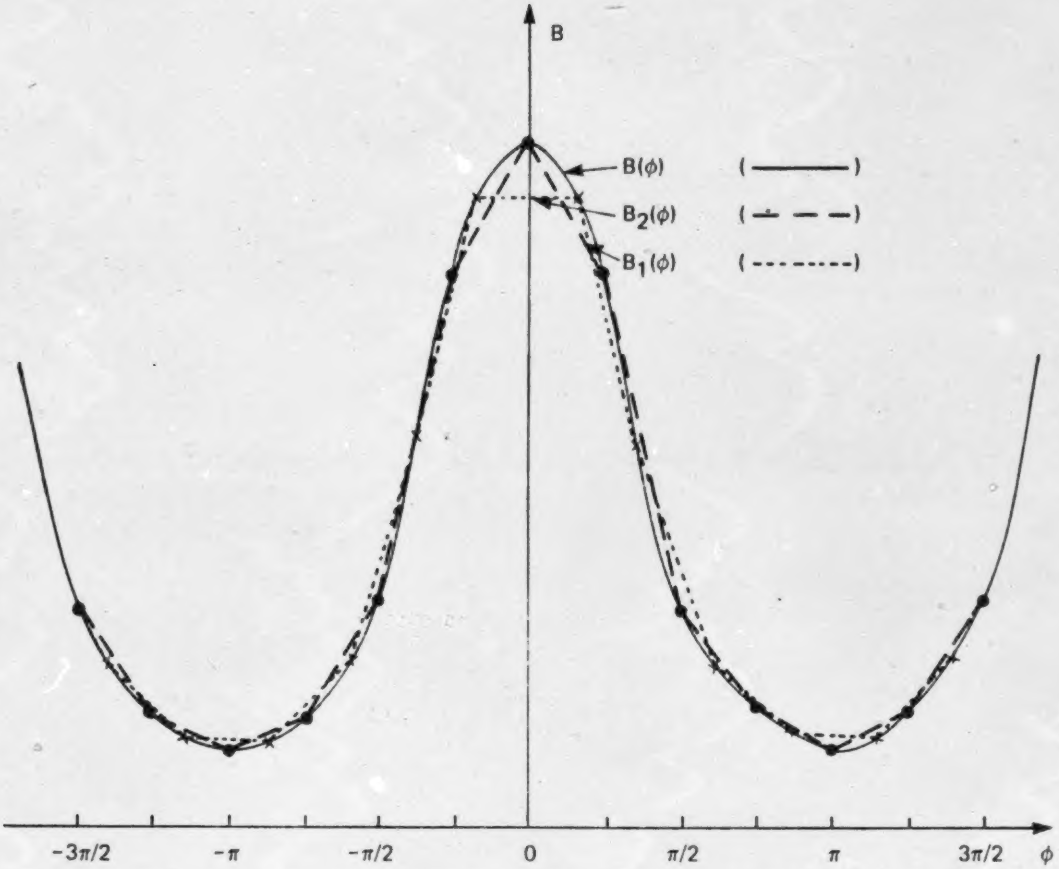


Figure A-7 – Precise and Approximating Brightness Functions

The maximum errors between $B(\phi)$ and $\hat{B}_2(\phi)$ occur at $\phi = 0$ and $\phi = \pm\pi/2$. The brightness error at $\phi = 0$ is given by

$$e_b(0,n) = B_S - B_S \cos\frac{\pi}{n}$$

$$e_b(0,n) = B_S\left(1 - \cos\frac{\pi}{n}\right) \qquad [A\text{-}12]$$

which has the same form as [A-7], the error in the surface approximation. The brightness error at $\phi = -\ \pi/2$ is maximum when $A_2 = 0$ resulting in constant brightness across the dark side of the object, Figure A-8.

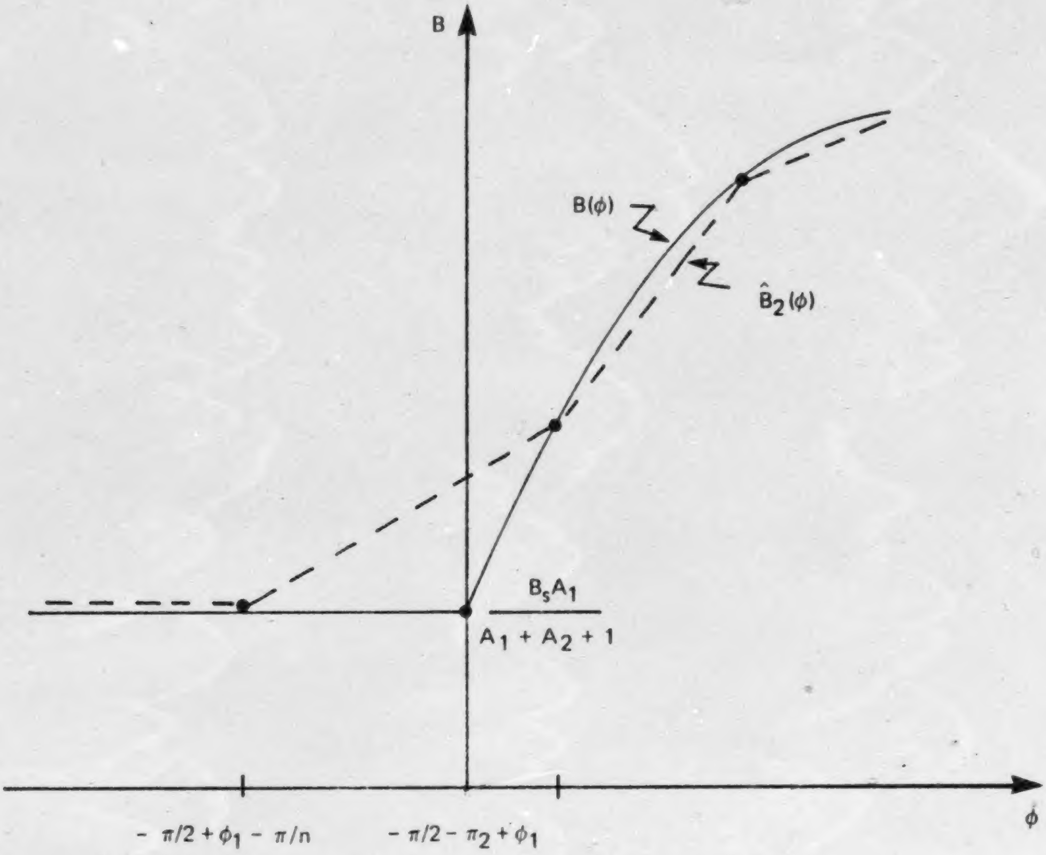


Figure A-8 – Brightness Error $e_b(-\pi/n, n, \phi_1)$

The brightness error at $\phi = -\pi/2$ for a phase angle of ϕ_1 radians is given by

$$e_b(-\pi/2, n, \phi_1) = \frac{\sin\phi_1}{(\pi/n)}\left(\frac{\pi}{n} - \phi_1\right) \qquad \text{[A-13]}$$

where $\phi_1 \in [0, \pi/2n]$. Since e_b is zero at $\phi_1 = 0$ and $\phi_1 = \pi/2n$, the maximum value of $e_b(-\pi/2, n, \phi_1)$ can be found from

$$\frac{\partial e_b}{\partial \phi_1} = 0 = -\sin\phi_1 + (\pi/n - \phi_1)\cos\phi_1 \qquad \text{[A-14]}$$

or

$$\tan\phi_1 + \phi_1 = \pi/n$$

For n sufficiently large $\tan\phi_1 \doteq \phi_1 \doteq \pi/2n$. For example, if $n = 8$, $\phi_1 \doteq \pi/2(8.1)$. Thus

$$e_b(-\pi/n, n) \doteq \frac{\sin \pi/2n}{2} \qquad n \geqslant 8 \qquad \text{[A-15]}$$

Table A-1 lists values for $e_b(o,n)$ and $e_b(-\pi/2, n)$ for various values of n. As expected $e_b(-\pi/n, n) > e_b(o,n)$ since the Taylor series expansion for $e_b(-\pi/n, n)$ is dominated by $(\pi/n)^2/2!$, while the Taylor series expansion for $e_b(o,n)$ is dominated by the smaller $(\pi/n)^3/3!$.

n	$e_b(-\pi/n, n)/B_S$	$e_b(0,n)/B_S$
8	.19134	.07612
9	.17101	.06031
10	.15451	.04894
20	.07821	.01231

Table A-1. **Brightness Approximation Maximum Errors**

The human eye is also quite sensitive to the rate of brightness change. The derivative of the brightness functions of Figure A-7 is shown in Figure A-9. The derivative of the true brightness function $B'(\phi)$ has discontinuities only at $\phi = \pm\pi/2$ in $\phi \in [-\pi, \pi]$; this discontinuity occurs on the locus of points separating the sunlit side of S from the side of S illuminated only by diffuse light. Hence, that discontinuity serves a very useful purpose to the eye.

It can be noted from Figure A-9 that $B'_1(\phi)$ approximates $B'(\phi)$ more closely than $B'_2(\phi)$. Both $B'_1(\phi)$ and $B'_2(\phi)$ converge to $B'(\phi)$ as n becomes large.

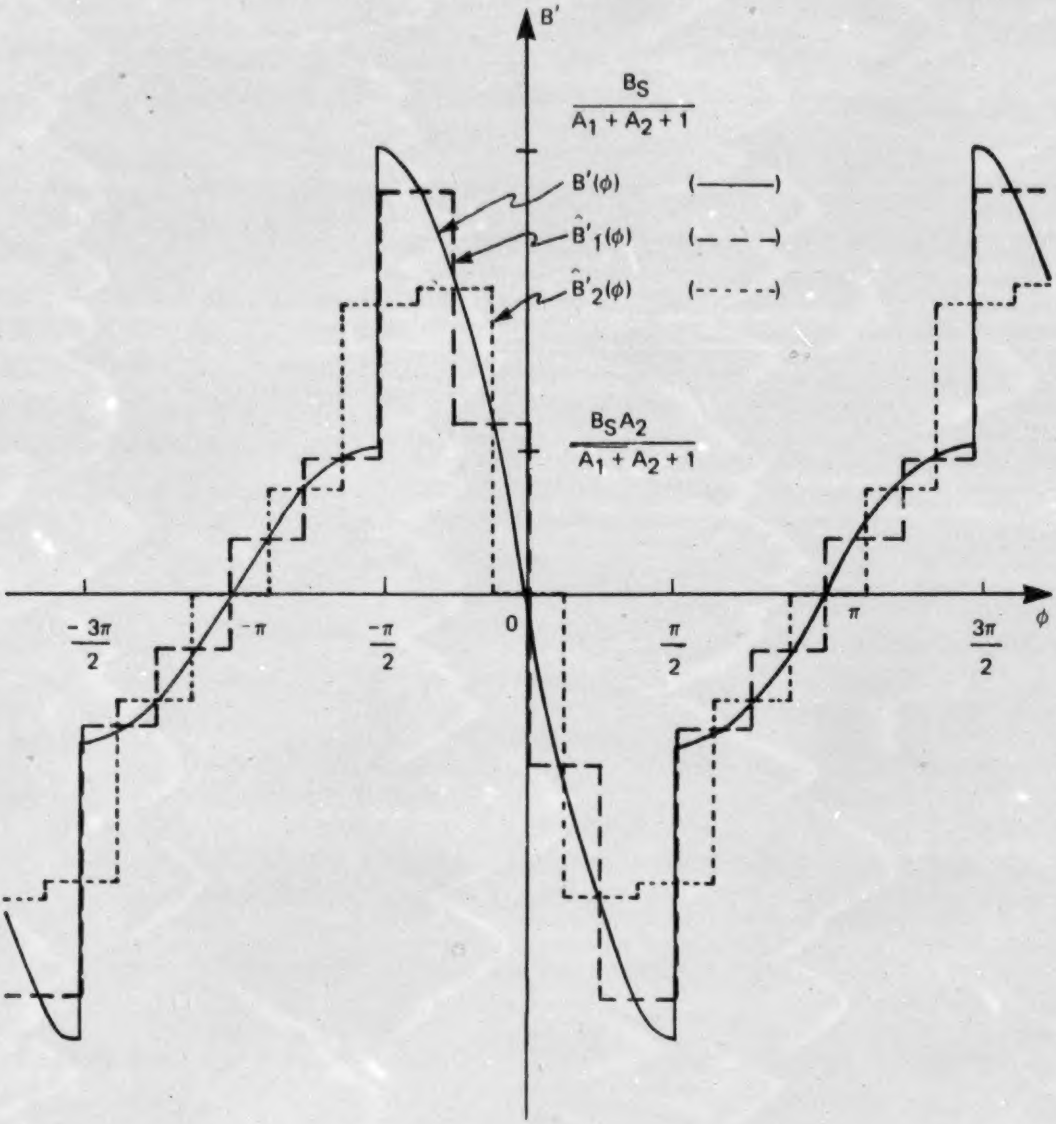


Figure A-9 – **Brightness Function Derivatives**

The same general conclusions occur if the logarithmic rather than the linear smooth-shading rule is utilized.

Linear Rule: $B(\lambda) = B_o + \lambda\,(B_f - B_o) \quad \lambda \in [0,1]$ [A-16]

Logarithmic Rule: $\hat{B}(\lambda) = B_o(B_s/B_o)^{\lambda} \quad \lambda \in [0,1]$ [A-17]

The deviation of the two approximations is given by

$$\begin{aligned} D(\lambda) &= B(\lambda) - \hat{B}(\lambda) \\ &= B_o + \lambda\,(B_f - B_o) - B_f (B_f/B_o)^{\lambda} \qquad \text{[A-18]} \end{aligned}$$

and the normalized deviation is given by

$$Dn(\lambda) = \frac{D(\lambda)}{B_o} = 1 + \lambda \left(\frac{B_f}{B_o} - 1\right) - \quad B_f/B_o)^{\lambda} \qquad \text{[A-19]}$$

The value of λ for which $D_n(\lambda)$ is maximum can be obtained by setting $\partial D_n(\lambda)/\partial \lambda = 0$ and is given by

$$\lambda_{max} = \frac{\ell n\,(B_f/B_o - 1) - \ell n\,(\ell n\,(B_s/B_o))}{\ell n\,(B_f/B_o)} \qquad \text{[A-20]}$$

and so $\lambda : ½ \rightarrow 1$ as $(B_f/B_o) : 1^{+} \rightarrow \infty$ as shown in Figure A-10. Figure A-11 demonstrates the deviation of the approximations for $(B_f/B_o) = 2$ and $(B_f/B_o) = 10$.

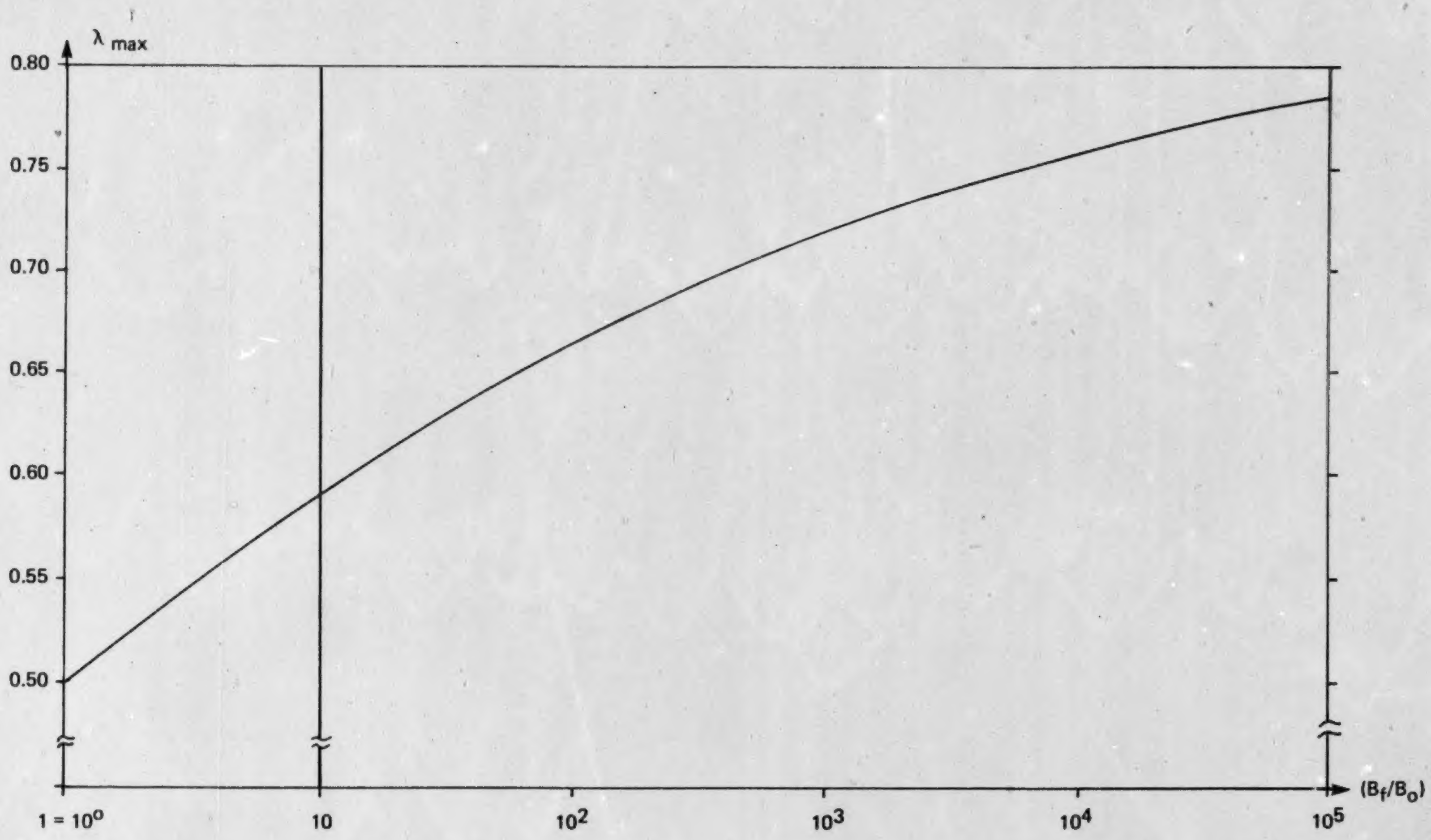


Figure A-10 – **Variation of λ_{max} With (B_f/B_o)**

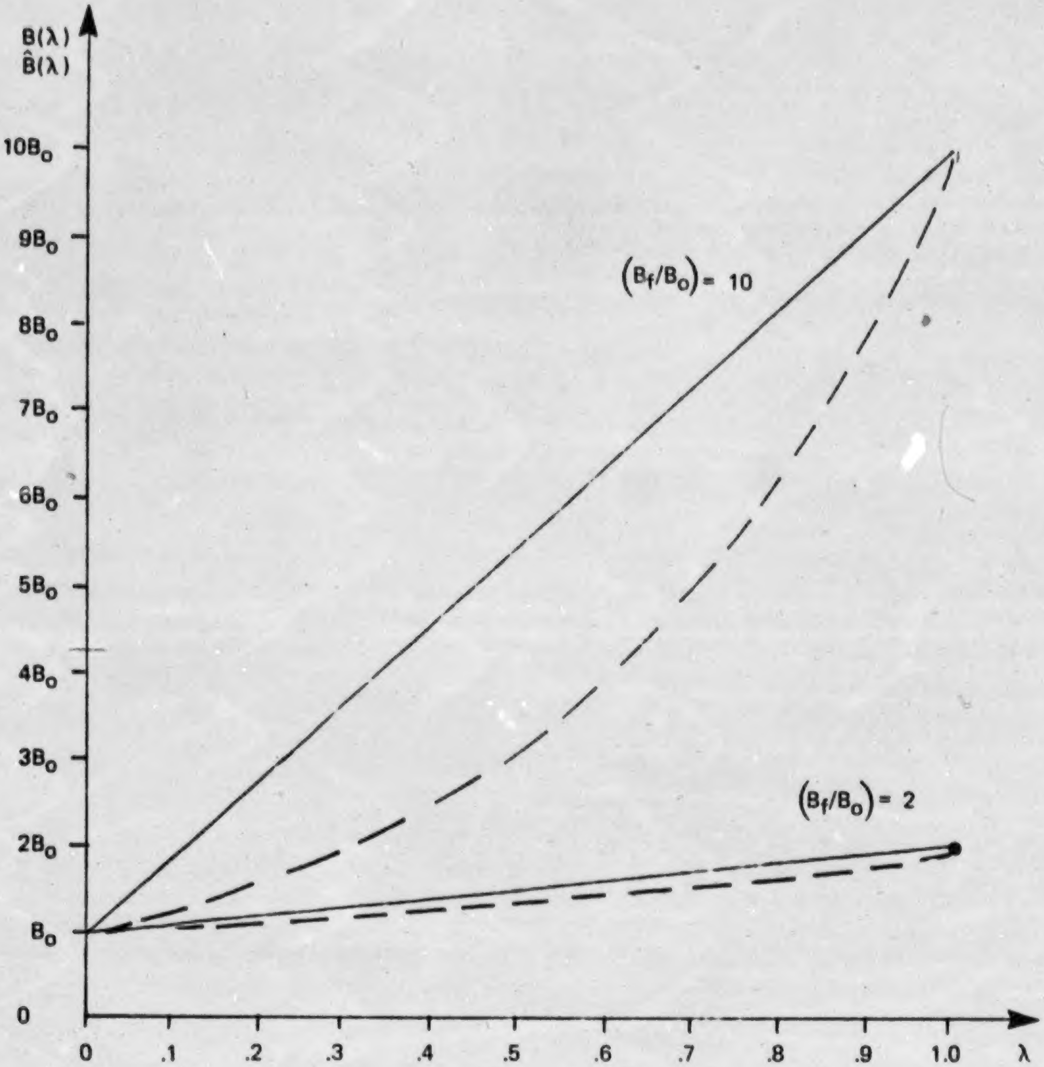


Figure A-11 – **Deviation Between B(λ) and $\hat{B}$ (λ)**

Appendix B

LOGARITHMIC BRIGHTNESS PARAMETERS

The problem here is to logarithmically encode a brightness $b \in [B_{min}, B_{max}]$ into an m–bit binary number. Here, $B_{max} > B_{min} > 0$. By the logarithmic rule

$$B = B_{min} \left(B_{max}/B_{min}\right)^{\lambda} \qquad [B.1]$$

for some $\lambda \in [0,1]$. Then

$$\frac{B}{B_{min}} = \left(\frac{B_{max}}{B_{min}}\right)^{\lambda} \qquad \lambda \in [0,1] \qquad [B.2]$$

and taking logarithms on both sides of [B.2] yields

$$\log_{\left(\frac{B_{max}}{B_{min}}\right)} \left(\frac{B}{B_{min}}\right) = \lambda \qquad \lambda \in [0,1] \qquad [B.3]$$

and so

$$(2^m - 1) \log_{\left(\frac{B_{max}}{B_{min}}\right)} \left(\frac{B}{B_{min}}\right) = b \qquad b \in [0, 2^m - 1] \qquad [B.4]$$

From a computational point of view

$$b = \frac{(2^m - 1)}{\ln\left(\frac{B_{max}}{B_{min}}\right)} \ln\left(\frac{B}{B_{min}}\right) \qquad b \in [0, 2^m - 1]$$

$$b = \frac{(2^m - 1)}{[\ln B_{max} - \ln B_{min}]} [\ln B - \ln B_{min}] \qquad b \in [0, 2^m - 1] \qquad [B.5]$$

or simply

$$b = K_1 \ln B - K_2 \qquad b \in [0, 2^m - 1] \qquad [B.6]$$

If $B_{min} = 1$, [B.5] reduces to

$$b = \frac{(2^m - 1)}{\ln B_{max}} \ln B \qquad b \in [0, 2^m - 1] \qquad [B.7]$$

Typical values are $B_{min} = 1$, $B_{max} = 149$, $m = 7$ and so

$$b = \frac{127}{\ln 149} \ln B \qquad b \in [0, 127] \qquad [B.8]$$

Appendix C

ERRORS IN VISIBLE EDGE POSITION AND BRIGHTNESS AT TERM

The CHARGE image generator uses 4-byte floating point representation for M—the slope of the visible edge, and g—the gradient of brightness along the edge. The CHARGE terminal is sent lower resolution representations of the slope and brightness gradient given by

$$M^* = \text{tr}\left[M \cdot 2^{11-p} + .5\right] 2^{p-11} \qquad \text{[C-1]}$$

$$g^* = \text{tr}\left[g \cdot 2^{11-p} + .5\right] 2^{p-11} \qquad \text{[C-2]}$$

where the operator tr [·] indicates truncation to the next lower or equal integer. In both [C-1] and [C-2], p is the number of high-order zeroes in an 11-bit fixed-point representation for the edge height, H. Since in general $M \neq M^*$ and $g \neq g^*$, there is generally a discrepancy between the brightness and horizontal position at the CHARGE image generator and at the CHARGE terminal. In particular, since edges must not cross each other at the CHARGE terminal, allowances for the slope error must be made at the image generator. If two edges approach each other to within the possible error amount, one must be terminated.

The horizontal error of any one edge at a distance Δy from the top of the edge is given by

$$e_x(\Delta y) = x_t + M \cdot \Delta y - (x_t + M^* \cdot \Delta y)$$

$$e_x(\Delta y) = (M - M^*)\Delta y \qquad \text{[C-3]}$$

So $|e_x(\Delta y)|$ attains it's maximum value when $\Delta y = H$.

Hence

$$e_x \leqslant (M - M^*)\, H = \left(\frac{\Delta x}{H} - \text{tr}\left[\frac{\Delta x}{H} 2^{11-p} + \frac{1}{2}\right] 2^{p-11}\right) H$$

$$e_x \leqslant \Delta x - \text{tr}\left[\Delta x + 2^{p-12} H\right] \qquad \text{[C-4]}$$

and so

$$|e_x| \leqslant |2^{p-12} H| \qquad \text{[C-5]}$$

If p = # high-order zeroes in an 11-bit H, then $H < 2^{11-p}$, and so

$$|e_x| < 0.5 \qquad \text{[C-6]}$$

where 1 = 1 horizontal discrete x – resolution interval (1/1600 of TV width)

Hence, since $e_x \in (-0.5, 0.5)$, two edges should not be allowed to come within one x-unit of each other at the CHARGE image generator to avoid the danger of crossover at the CHARGE terminal. If a minimum separation of 1 horizontal resolution unit is desired at the CHARGE terminal, the closeness criterion at the CHARGE image generator must be 2 units.

A similar analysis for the brightness gradient yields a maximum brightness error of

$$|e_b| < 0.5 \qquad [C\text{-}7]$$

where 1 = 1 brightness resolution interval (1/128 of brightness range).

Appendix D

CHARGE COMMANDS

Each CHARGE command falls into one of the following categories:

1. Atom commands—used by an author to create different 3-dimensional shapes (atoms).
2. Object commands—used by an author to define complex objects in terms of atoms and previously defined objects.
3. Viewing commands—used by an author to define parameters to be used by the image generator when displaying a collection of objects.
4. Image commands—used by an author to store and retrieve the results of an image generation process.

Commands may be entered in a free-form format with blanks separating the various components of each command and the character "/" serving as a delimiter between successive commands.

ATOM COMMANDS:

(1)	AT/	Invoke atom command processor.
(2)	C Atom/	Create Atom.
(3)	C Atom 1 Atom 2/	Create Atom 1 using Atom 2.
(4)	M Atom/	Modify Atom.
(5)	X Atom/	Scratch Atom from uncompiled Atom file.
(6)	LA/	List atoms in uncompiled Atom file.
(7)	U/	Compile atom currently being modified, and put it into compiled Atom file.
(8)	A (cm) . . . /	Add atom components: Point: $P(\#,x,y,z)$ Surface: $S(\#,P_1,P_2,P_3)$ Brightness Normal: $B(\#,S_1,S_2 \ldots,S_{10})$ Edge: $E(\#, \text{Type}, P_1,P_2,S_L,S_R,B_1,B_2,B_3,B_4)$
(9)	D (cm) . . . /	Delete atom components.
(10)	LP/	List all atom points.
(11)	LB/	List all atom brightness normals.

ATOM COMMANDS (Cont.):

(12)	LS/	List all atom surfaces.
(13)	LE/	List all atom edges.
(14)	LP n_1 - n_2/	List atom points n_1 to n_2.
(15)	LB n_1 - n_2/	List atom brightness normals n_1 to n_2.
(16)	LS n_1 - n_2/	List atom surfaces n_1 to n_2.
(17)	LE n_1 - n_2/	List atom edges n_1 to n_2.
(18)	AL/	Invoke object command processor.
(19)	IL/	Invoke image library command processor.

OBJECT COMMANDS:

(1)	AL/	Invoke object command processor.
(2)	C Object/	Create object.
(3)	M [Object]*/	Modify object.
(4)	S [Object] /	Show object.
(5)	X [Object] /	Scratch object.
(6)	A Object [tr] /	Add sub-object with transformations:
		m(x,y,z): Move in x,y,z
		S(x,y,z): Scale in $x,y,z > 0$
		X(x,y,z,θ_X): Rotate in X about (x,y,z)
		Y(x,y,z,θ_y): Rotate in Y about (x,y,z)
		Z(x,y,z,θ_z): Rotate in Z about (x,y,z)
		OR(P): Optical reflectivity $P \in [0,1]$
		C(b,g,r): Color if not yet colored.
		RC(b,g,r): Color
		MIX: Mirror image in X.
		MIY: Mirror image in Y.
		MIZ: Mirror image in Z.
(7)	D [object[.#]] . . . /	Delete sub-object.
(8)	I Object [.#] # [-#] [tr] . . ./	Insert transformations.
(9)	R Object [.#] # [-#] [tr] . . ./	Replace transformations.
(10)	O [Object] /	List object structure.
(11)	T [Object 1 [Object 2]] /	List transformations.
(12)	N Object 1 Object 2/	Name Object 1 Object 2.

*An argument in brackets denotes an optimal argument.

OBJECT COMMANDS (Cont.)

(13)	L [#]/	List object names.
(14)	LA/	List atom names.
(15)	UC Object/	Show use count of object.
(16)	AT/	Invoke atom command processor.
(17)	IL/	Invoke image library command processor.
(18)	ZZ/	Drop CHARGE system.

VIEWING COMMANDS: (processed by object command processor)

(1)	V [par] . . ./	Set viewing parameters.
(2)	VR [par] . . ./	Reset viewing parameters to their default values and then set parameters: W(x,y,z): Viewing window location. E(x,y,z): Eye location with respect to window. O(x,y,z): Point observed. ws(x,y): Window size in meters. MAG(M): Magnification $M > 0$. FS(F): F-stop for observer $F \in [0,1]$. BR(B): Maximum brightness at terminal. RES(R_x,R_y): Resolution of terminal. SUN(x,y,z): Sunlight direction. SB(.5): Fraction of B for sun. BC (b,g,r): Background default color. BB(b): Background default brightness. AB(a_1,a_2): Ambient brightness factors. T(t): Terminal type: 1 : normal 2: no smooth-shading

IMAGE LIBRARY COMMANDS:

(1)	IL/	Invoke image library command processor.
(2)	SU Image/	Save Image.
(3)	SH Image/	Show Image.
(4)	LI/	List image names.

IMAGE LIBRARY COMMANDS (Cont.)

(5)	SN [#] . . ./	Show image(s) by number.
(6)	DI Image . . ./	Delete Image.
(7)	RI Image 1 Image 2/	Rename Image 1 Image 2.
(8)	CP Packet [Image] . . ./	Create image packet.
(9)	DP Packet/	Delete packet.
(10)	LP Packet/	List packet contents.
(11)	SP Packet/	Send packet.
(12)	RP Packet 1 Packet 2/	Rename Packet 1 Packet 2.
(13)	RS/	Resend last image.
(14)	AT/	Invoke Atom command processor.
(15)	AL/	Invoke Object command processor.